Design • Build • Program • Learn • Compete

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

March 2012

A CNC Machine For Roboteers

Can You Build One? I Did!

What could you build with it?

✦ Sounding Off

Give your Arduinobot electronic sound effects, music, and even voice.

✦ Geerhead

Robotic Venus Flytrap demonstrates new technology for medical applications.

U.S. $5.50 CANADA $7.00

03

0 71486 02422 4

GOT A QUESTION? GET ON THE HORN!

Remember the good old days when you were able to speak to a live person and ask them a question? You still can at Hitec, where we provide the best of both worlds: Old fashioned customer service with modern technology! Our legendary service department provides expert advice and quick, dedicated solutions to all your robitic inquiries. From recommending the perfect servo for your vehicle to providing instruction about our line of battery chargers, the dependable and caring Hitec staff is here for you.

Hitec Customer Service

Monday thru Friday

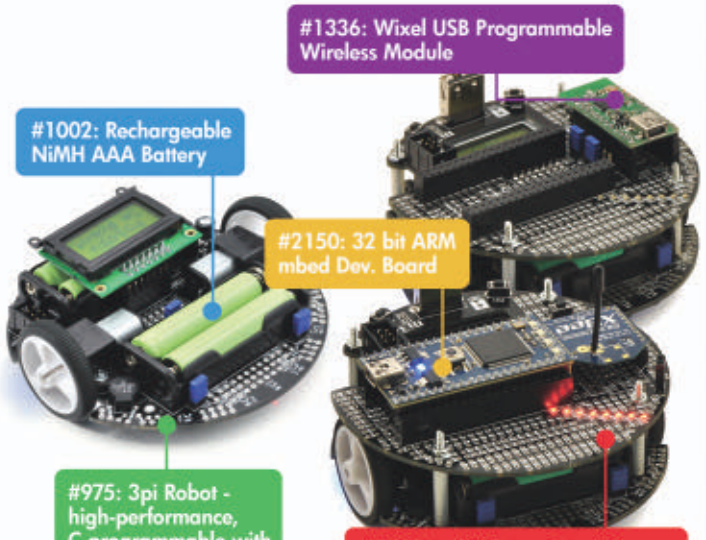
8:00am – 4:30pm Pacific Standard Time

Expect More! Expect Hitec!

12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

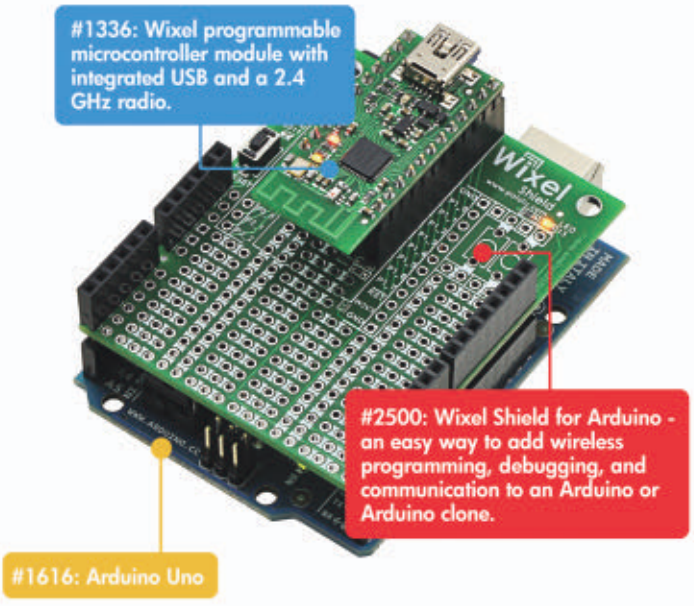
Robots and Robot Kits:

Pololu 3pi and m3pi



Programmable Controllers:

Wixel and Wixel Shield

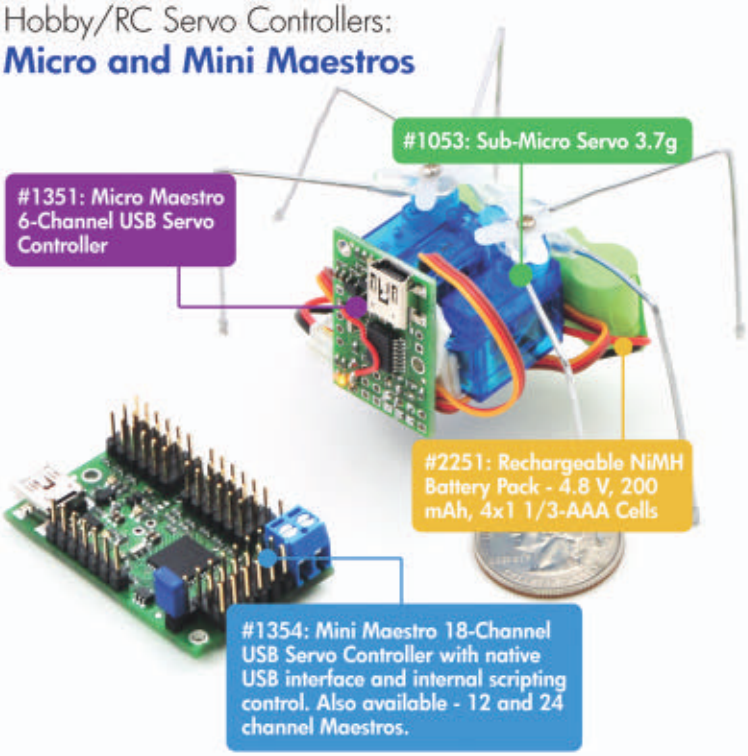


Sensors:

Orientation, Reflectance and More

Hobby/RC Servo Controllers:

Micro and Mini Maestros



Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products - from actuators to wireless modules - that can help you take your robot from idea to reality.

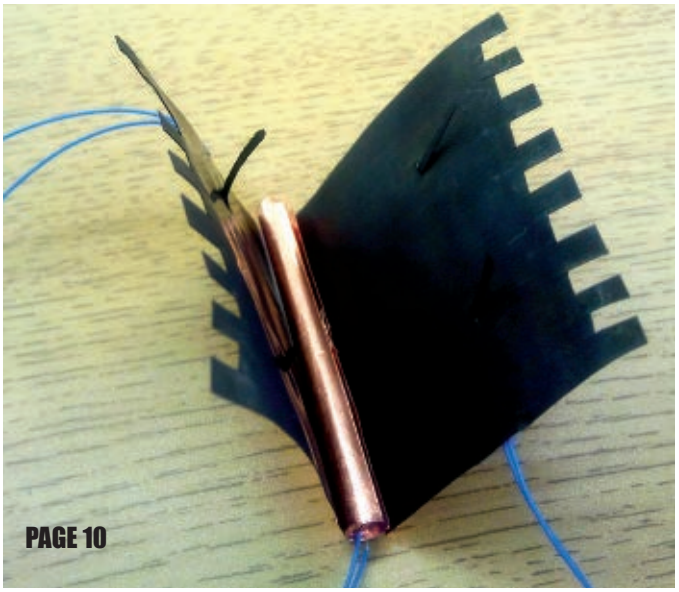
Pololu
Engage Your Brain

Find these products and more at **www.pololu.com**

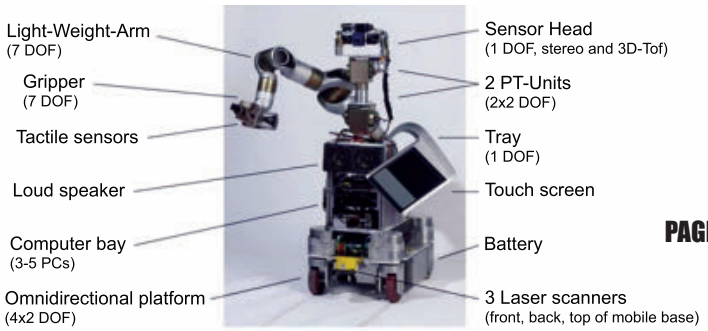
SERVO MAGAZINE

03.2012

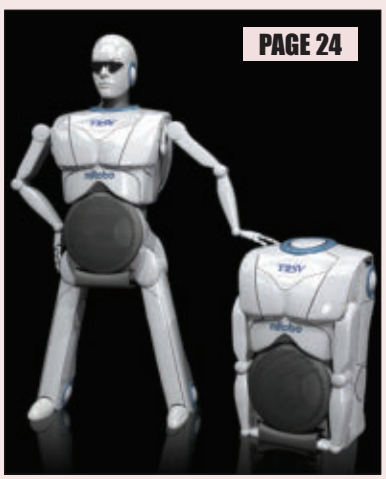
VOL. 10 NO. 3



Columns

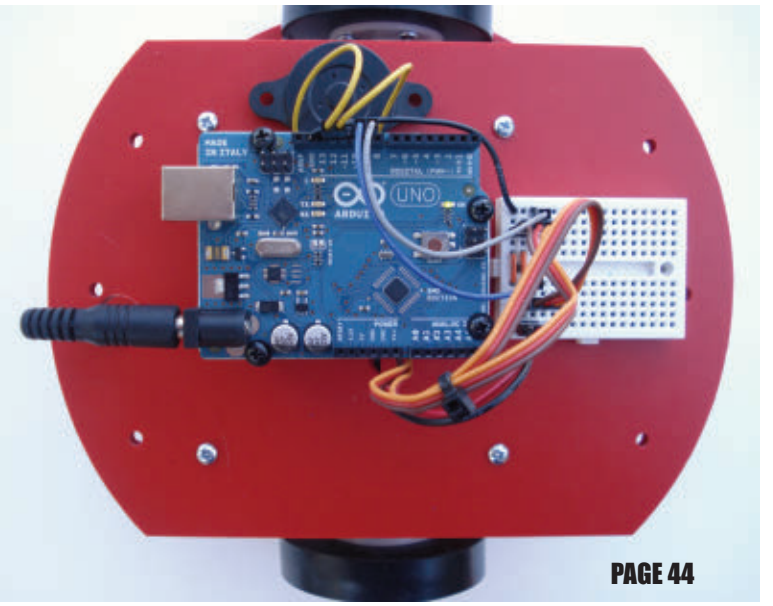


Departments

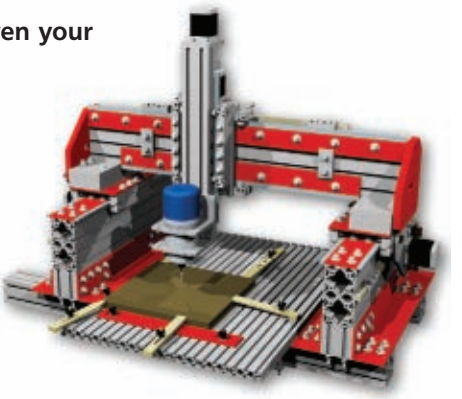


SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



PAGE 44



The Combat Zone...

Features

Events

Mind / Iron

by Bryan Bergeron, Editor

What's The Point?

I just purchased a pair of those R/C mini-helicopters for about $20 each. Consider that even if I could design such a craft, the parts for a remote control, battery recharging circuit, and radio receiver would cost more than $20. The helicopter body, servo motors, gyroscope, and gears would probably cost another $200, assuming I could find the parts and the plans to build the helicopter.

This reality is both cause for celebration and potential depression. On one hand, robotics has never been as affordable for the roboticist. If you're looking for raw materials for your project, it's simple enough to buy a fully functional device and strip it for parts. Stripping one of those helicopters yielded dozens of small parts that I couldn't have located — much less made on my own. (Perhaps I could have made a few of the parts if I had one of those 3D printers.)

On the other hand, what's the point of scuttling a perfectly good R/C craft? After all, why not just use the craft as a platform and try to attach a micro-camera? That's a reasonable approach, and one that would allow you to focus on, say image recognition, as opposed to getting a micro-camera up in the air.

However, avoiding platform development is probably shortsighted. There's a lot to be learned from building your own robotics platform. Even if your creation is heavier, bulkier, more expensive, and less maneuverable than what's commercially available, there is the fringe benefit of learning how to engineer a platform on your own. That might translate to frequent crashes, delays, and some degree of frustration. However, in the long term, you'll know not only how to flip a power switch, but how to repair, upgrade, and modify your creation.

I'm not suggesting you ignore all of the great bargains out there. Instead, leverage what you can get your hands on by performing non-destructive teardowns. Working incrementally, disassemble and then reassemble the device. For example, if you have one of those mini-helicopters, start with the rear rotor. How is it wired? How large and how powerful is the motor? Are the blades fitted directly on the motor or through gears?

Next, examine the power supply. Use your DMM and a power resistor to determine the voltage and milli-amp-hour rating of the Lithium battery. How is the voltage regulated during flight? Is there anything onboard to protect the battery against overcharging? What can you learn from the charging station? What safety circuitry or physical barriers are in place to prevent overcharging or reversing the polarity of the charge? In short, what are the takeaway lessons that you can apply to your own designs?

Everyone loves a bargain. There is no rule that says you can't enjoy a few toys while you're experimenting with robotics. If you want to get a real feel for what it takes to create a robot however, you have to take out your tools, bag of parts, schematics, and get to work. After all, robotics is about the journey. **SV**

BIO-FEEDBACK

Dear *SERVO*:

I'm writing to you because I wasn't the only author of the Pathfinder Rover article that was in the February '12 issue. It was also written by my project colleague Rosia Nicolae and by our teacher (project advisor) Molder Cristian. Rosia and I are students at the Military Technical Academy from Bucharest, Romania. Thank you!

Chifor Bogdan

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert
Tom Carroll
Dennis Clark
Kevin Berry
Alan Federman
Thomas Kenney
Morgan Berry
Evan Woolley
Jenn Eckert
David Geer
R. Steven Rainwater
Gordon McComb
Michael Simpson
Mike Jeffries
Bryce Woolley

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

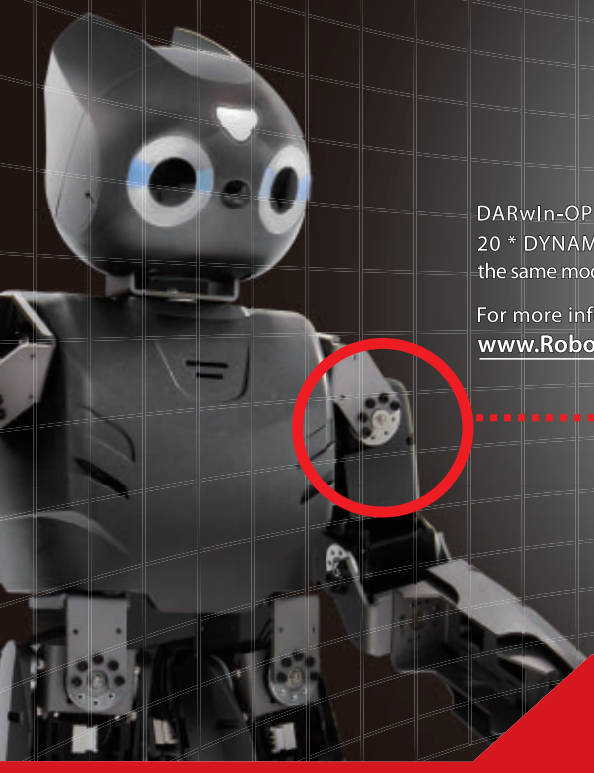
MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

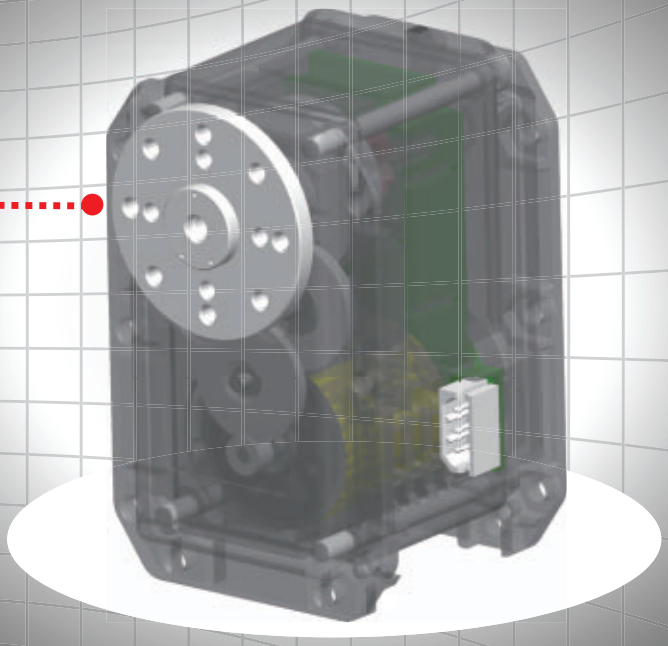
PRODUCTION/GRAPHICS
Shannon Christensen

What's New?



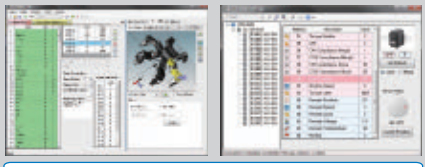
DARwIn-OP consists of
20 * DYNAMIXEL MX-28,
the same model for all joints.

For more info, please visit
www.RobotSource.org



DYNAMIXEL MX-Series

S/W Tools



MATLAB® LabVIEW Microsoft Visual Studio python Java
C/C++ C# Visual Basic

Contactless Absolute Encoder

12-bit resolution, 360° operating range
with super-durability

User Programmable PID Control Gain

PID control

Precise and reliable PID control

Powerful Resource & Organized Design

360° 4096 steps

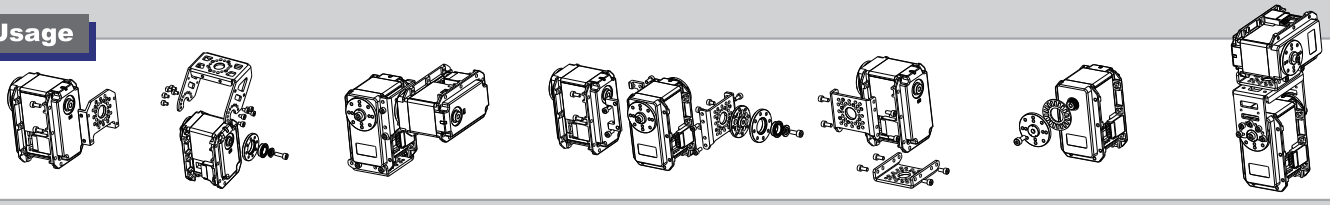
Maxon motor, 32-bit controller,
high communication speed
RX and EX series compatible dimension

Current Based Torque Control

Torque

Position and speed control
with dual-loop current control
(MX-64, MX-106 only)

Usage



[USA] E-mail : america@robotis.com Tel : +1-949-333-3635
[KOREA] E-mail : korea@robotis.com Tel : +82-70-8671-2604
[JAPAN] E-mail : japan@robotis.com Tel : +81-3-4330-3660
[Other Countries] E-mail : contactus2@robotis.com Tel : +82-70-8671-2609

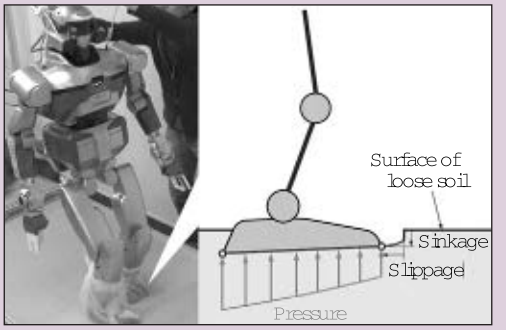
ROBOTIS
www.robotis.com


Robytes

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com.

by Jeff and Jenn Eckert

Botprints in the Sand

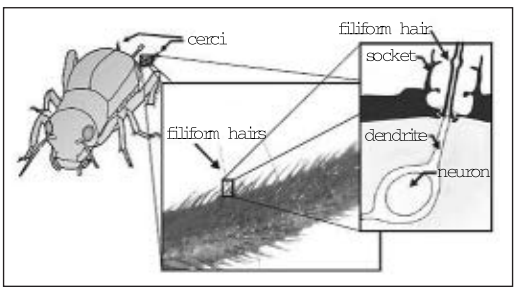


Test setup showing slippage and sinkage when a robot walks on sand.

You probably have never seen bipedal robots strolling down Waikiki Beach or hiking across the Sahara desert, and for good reason. Humans can trudge through sand pretty well, even after a few piña coladas, but bots aren't good at walking on a medium that shifts, slips, and allows them to sink in. Apparently, doing so tends to discombobulate their balance systems when data from the accelerometers indicate an unsteadiness in the feet.

Until recently, most studies of robotic gaits have assumed stiff ground underneath, so there has been no basis for designing droids that can walk over mutable surfaces. However, at last year's International Conference on Flow Dynamics held in Sendai, Japan, researchers from Tohoku University's (**www.tohoku.ac.jp**) Graduate School of Engineering presented a paper providing an analysis of "slip-sinkage characteristics of sand as a fundamental research for realization of biped walking of a humanoid robot on sand."

The next phase will be to come up with control software to deal with different types of sand or loose soil. A practical implementation, of course, will also have to address things like salt corrosion and the effects of sand in precision mechanics, but who knows? Someday the bully who kicks sand in your face may be even less human.

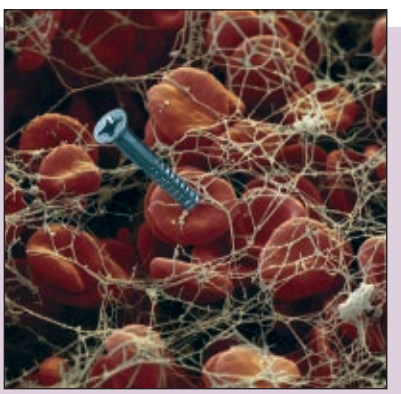


Layout of a cricket's abdominal sensory system.

Synthetic Cricket Belly

There seems to be no end to the creature characteristics that can be mimicked for robotic purposes, and one of the latest comes from the cricket — a familiar little guy with some odd characteristics. Not only do crickets have their ears in a strange location (just below the knees on their front legs), they also have tiny, sensitive abdominal hairs that enable them to detect an approaching predator and accurately estimate its distance and direction. These whisker-like appendages — known as cerci — have inspired researchers at the Netherlands' University of Twente (**www.utwente.nl**) to come up with a sensor that is likewise highly sensitive to airflow. The synthetic hairs are made of polymer SU8 which is a photo-imageable epoxy developed a long time ago by IBM. Each hair is 0.9 mm long and slightly thicker at the base than the top. These are embedded in a flexible microsystem that registers even very tiny movements via capacitance changes. Interestingly, the sensitivity of the hairs can be tuned to a specific frequency range simply by electronically adjusting their spring stiffness. Intended applications include robotic sensors and studies of very specific airflows, but no details were offered.

Screw the Clots



Helical treatment of a blood clot (cheesy dramatization).

The concept of sending microbots into your body to treat various maladies isn't particularly novel, but a new "twist" on the concept has been described by researchers from South Korea's Hanyang University (**www.hanyang.ac.kr**) in Seoul and Chonnam National University (**web.chonnam.ac.kr**) in Gwangju. At the recent Annual Conference on Magnetism and Magnetic Materials (Scottsdale, AZ), they described a new navigation system that drives 1 mm microbots throughout the human body to attack tumors, blood clots, and hard-to-reach infections. The system uses external magnetic fields to generate two types of motion: translational (side-to-side) and helical (corkscrew-like). The latter could allow the microbots to screw themselves into blood clots and other obstructions like a John Deere with an auger bit digging post holes. According to the inventors, the concept may be extended for "precise and effective manipulation of a microbot in several organs of the human body, such as the central nervous system, the urinary system, the eye, and others." Let's just hope they don't run amok in the cerebrum while the operator is on a coffee break.

The Kobot foldable robot scooter.

Urban Scooterbot Introduced

Last December, Japanese venture company Kowa-tmsuk (**kowatmk.co.jp**) used the Tokyo Motor Show as a launch pad for Kobot, described as a "foldable robot scooter." We're struggling to see what is all that robotic about it, but presume it refers to the driver's ability to use a smart phone to remotely tell the vehicle to fold its seat and rear wheel into the body. This results in a footprint of only a square meter which could be an advantage when parking in extremely crowded city streets. With a top speed of only 30 kph (18 mph), it seems more likely to cause traffic jams than to alleviate them. But, hey, it's sure cute. Shown is the single-seat configuration, but a two-seater is in the works. Company President Yoshito Serita observed, "They are unique vehicles designed to be super-small, super-zippy, and full of playful spirit." He also surmised that Kobots represent "the future of urban driving." Didn't we hear something like that about Segways, personal zeppelins, and rocket belts? **SV**

GEERHEAD

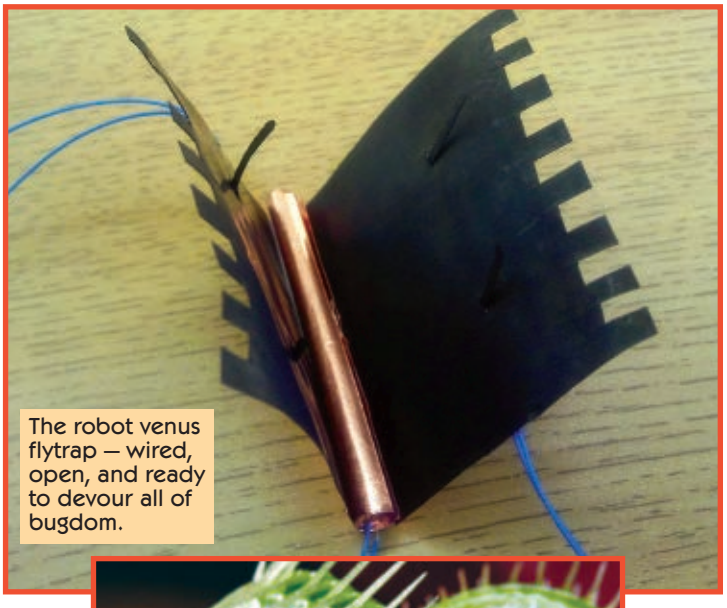
by David Geer

Contact the author at geercom@windstream.net

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Robotic Venus Flytrap Models Small Muscle Function; Catches Bugs To Use As Fuel

A prototype for a robotic venus flytrap has been developed, mainly to recreate small muscle function. The flytrap we'll discuss here uses polymer membrane muscles coated with gold or platinum electrodes that can respond to electric current to quickly close the flytrap's lobes or leaves on its insect prey. By applying polarity and then reverse-polarity, the robot flytrap closes on sensed bugs and then opens again. This kind of capability has potential for numerous medical applications. Patients suffering from facial paralysis require the use of many tiny muscles, and such soft robotic material could one day be implanted in the face to help. Similarly, people with eye and heart diseases could benefit from high performing muscle-like material.



The robot venus flytrap – wired, open, and ready to devour all of bugdom.

When a bug lands on the membrane, the membrane trigger hairs or bristles bend producing a current. Dr. Mohsen (Mo) Shahinpoor, PhD applies this current reaction as a sensor, informing the flytrap mechanism of the bug's presence and triggering a larger current through a solid-state relay that closes the robot flytrap. This robot venus flytrap could be considered for use as the mouth of the Ecobot III, to provide fuel for it to run on. Recall that the Ecobot contains a fully functional digestive system.

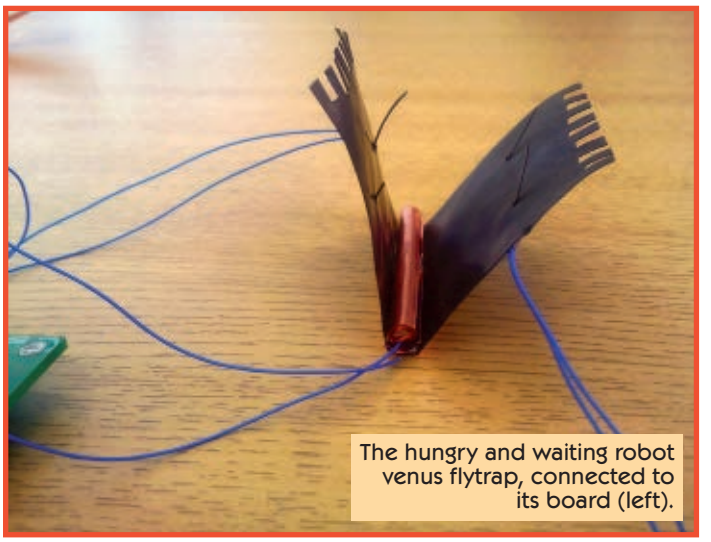
The robot venus flytrap is intended to mimic its biological counterpart. In nature, the real venus flytrap (Dionaea muscipula) is one of 500 different types of plant life that eat meat. It lays its cunning trap – delicious sap for the hungry spider or other insect – captures its prey physically in its leaves, and eats the bug meat.

Using the hairs on its lobes, the venus flytrap detects movement. If it senses brushing against more than one hair within a 20 second interval or against the same hair in much smaller intervals, it closes its lobes to catch its feast. Any other movement could be lifeless and coincidental, not providing the plant with anything good to eat.

If the venus flytrap has trapped a living creature, the prey will move around violently in an attempt to escape. Sensing this, the trigger hairs confirm that the object in the trap is indeed live meat the venus flytrap wants, and the leaves will close around the bug even tighter. If there is no thrashing, the leaves will eventually reopen.

Though you would expect to find them in the rain forest, these carnivorous plants which feed on slugs, caterpillars, and crickets (as well as arachnids and the flies that give it its name) are found in North and South Carolina. While the venus flytrap gets a lot of its nutrition from photosynthesis, it also needs nitrogen and other nutrients which it cannot get from the ground because the soil in the region is to acidic.

The venus flytrap has adapted to its surroundings to retrieve its fair share of potassium, phosphorus, calcium, sulfur, and magnesium, as well as nitrogen by digesting animal life. This clever plant both traps and digests its victims in the same leaves which basically form its stomach.



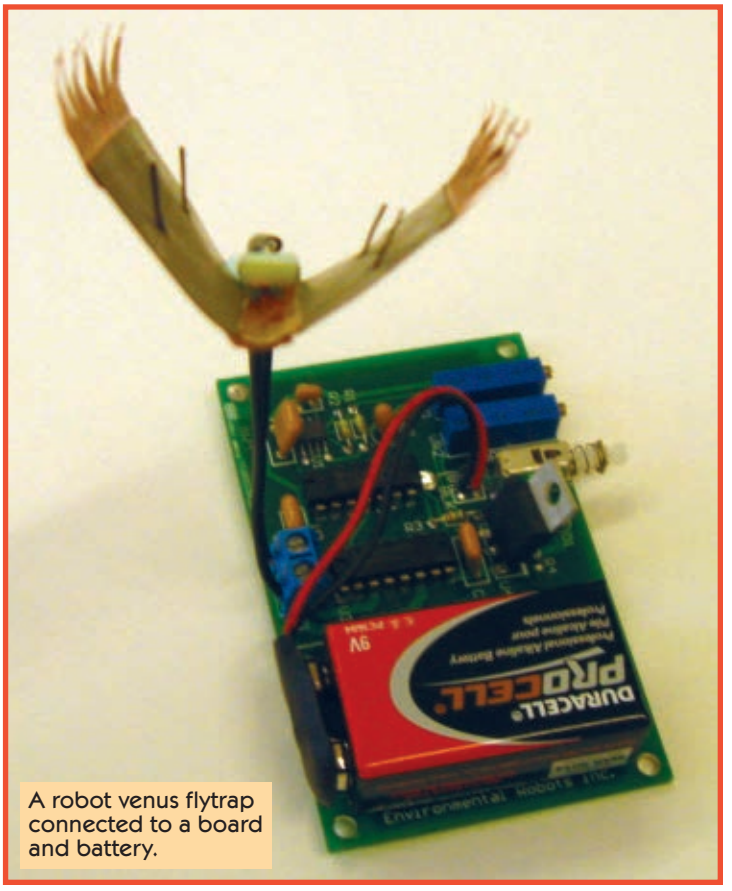
The hungry and waiting robot venus flytrap, connected to its board (left).

Robot Chow Time

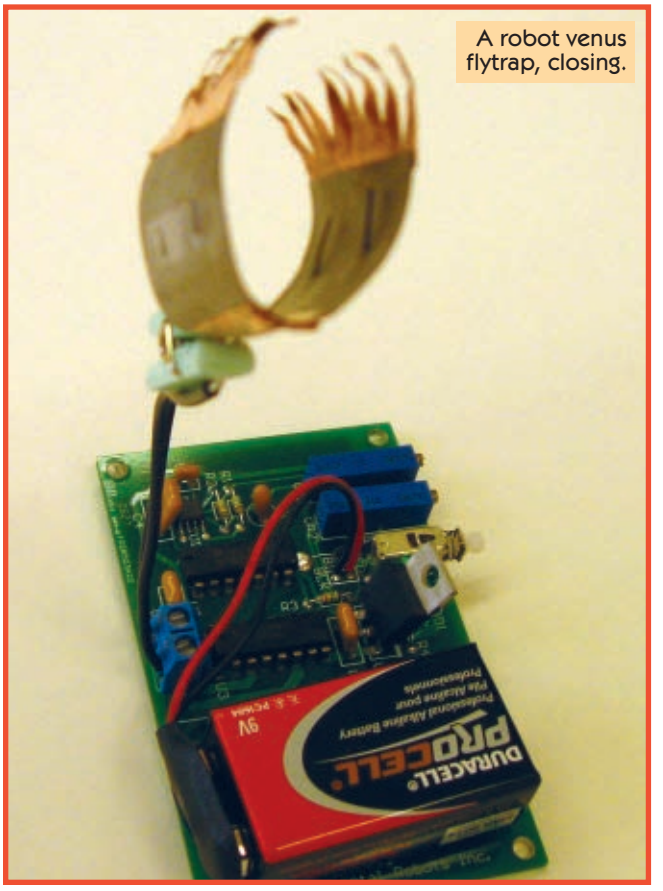
The robot flytrap's polymer lobes remain open in their initial or starting configuration. The lobes have bristle-like microfingers or trigger hairs made of ionic polymeric metal nanocomposite (IPMC) micro-muscles. Once the muscles trigger, wiggle, or move due to stimulus from some type of bug that has landed, these triggers generate a millivolt signal that passes to a relay.

When a fly or other prey sits on the open lobes, the microfingers generate enough voltage to activate a smart solid-state relay system that essentially switches on the actuation circuit to apply a larger voltage across the lobes. This causes the IPMC muscles to close tightly, encasing the insect in a mechanical tomb.

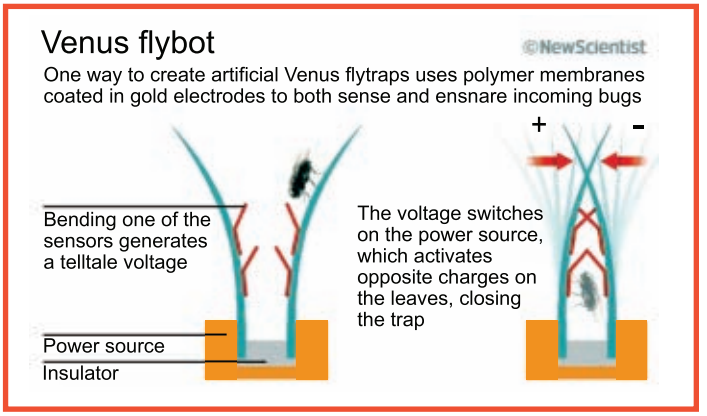
The inherent nature of the IPMC materials make the chain



A robot venus flytrap connected to a board and battery.



A robot venus flytrap, closing.



reaction possible. "The basic material of the IPMC is an ionic polymer or, technically, a polyelectrolyte or polymer electrolyte that dissociates into anions and cations when polarized by an ionic medium. The material turns into a nanocomposite through the application of a metal such as gold or platinum or a conductor such as carbon nanotube or graphene. An important criterion in selecting the material is the possible cations that the polymeric electrolyte can contain. A good example of a qualifying material is ionic Teflon together with platinum as a conductor and catalyst to enhance the force generation of the muscles," explained Dr. Shahinpoor.

The Nature of the Robotic Lobes and Leaves

A special procedure called redox (used to chemically plate the basic polymeric material) creates a robotic substance that is active in sensing and actuation everywhere in the material itself — down to a nanolevel in size. Thus, it is basically a robotic material that is a distributed nanosensor and nanoactuator. "For example, if you can cut it by laser or nano-cutters into small nanosize dimensions, it can still sense and actuate," says Dr. Shahinpoor.

The gold electrodes that Dr. Shahinpoor used create internal electrodes within and without the material for applying a voltage to the material for actuation, or to give a signal in voltage upon sensing or deforming the IPMC substance. "You can use any metal but for sustainability and some medical applications, gold or platinum are best," comments Dr. Shahinpoor.

The IPMCs generally bend toward the anode (+) electrodes because the imposed electric field moves the internal cations towards the cathode (-) electrode. "If you impose an oscillatory voltage on the IPMCs, these exhibit oscillatory vibrational movements back and forth as the polarity of electrodes are reversed in time with a given frequency," Dr. Shahinpoor continued.

Parts and Nano Parts

By placing a common electrode on the abside (upper side or trap side) of the traps in the middle or spine of the two lobes, it causes the lobes to bend toward the common electrode and move towards each other to close the trap on the prey. This works because the IPMC lobes bend toward the anode electrode as previously mentioned. The lobes — which move as if there are internal molecular motors at work in them — are referred to as distributed biomolecular motors themselves.

Dr. Shahinpoor and his colleagues manufacture their own IPMC materials and muscles in their smart materials and artificial muscle laboratory. They sometimes use Nafion from DuPont for the base material. "We buy gold or platinum or other metal compounds from the Sigma-Aldrich chemical company," he says.

The sensing circuit is simply a pair of electrodes connected to the bristle hairs. The signal feeds into a solid-state relay which can then switch on the actuation circuit, which uses an integrated signal generation circuit with a 555 timer and a voltage regulator chip.

In practice, researchers could apply a sweet sap or other chemical to the flytrap to attract prey. However, this has not been done yet in the course of their research.

Conclusion

Researchers like Dr. Shahinpoor are making progress toward self-sustaining robots that power themselves through nature, rather than relying on batteries and traditional power sources that are expensive, time limited, or create a burden with their carbon footprints. The Ecobot is a potential robot that could digest the bugs the venus flytrap captures and use them as fuel, without having to depend on other manmade power sources. **SV**

All photos are courtesy of the Biomedical Engineering lab at the University of Maine.

Resources

Robot venus flytrap press
http://umaine.edu/news/blog/2011/10/30/multiple-reports-on-shahinpoor-innovation/

DuPont Nafion material
www2.dupont.com/FuelCells/en_US/products/nafion.html

Sigma-Aldrich materials
www.sigmaaldrich.com/united-states.html

Dr. Shahinpoor, PhD, creator of robot venus flytrap
http://umaine.edu/mecheng/faculty-and-staff/mohsen-mo-shahinpoor-ph-d-p-e/

Dr. Shahinpoor's resume
www.umaine.edu/MechEng/mo/ShahinpoorMo-CV-Acad-11-10-11.pdf

The Ecobot
www.brl.ac.uk/projects/ecobot/index.html

MIT Press papers, including Ecobot III papers
http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=12433&mode=toc

Lynxmotion

The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!

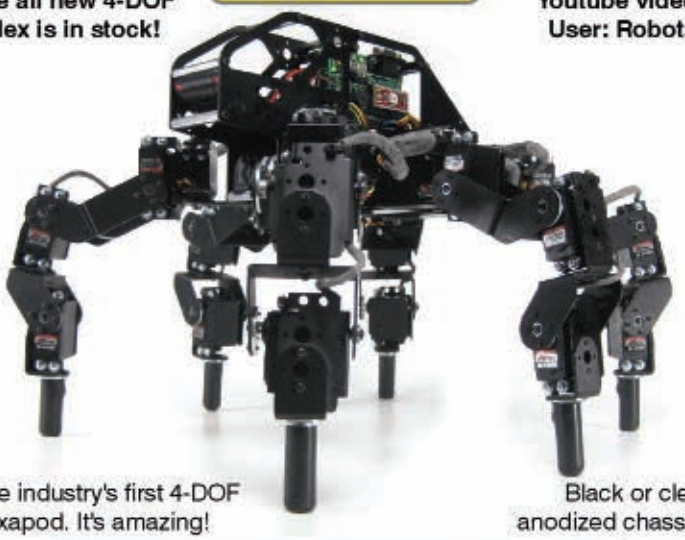
Biped Nick

Biped Pete

Featured Robot

The all new 4-DOF T-Hex is in stock!

Youtube videos User: Robots7



The industry's first 4-DOF hexapod. It's amazing!

Black or clear anodized chassis!

Biped Scout

Biped 209

Walking Stick

With our popular Servo Erector Set you can easily build and control the robot of your dreams!

Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.

All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.

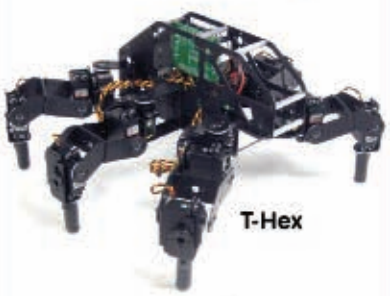
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.

SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!

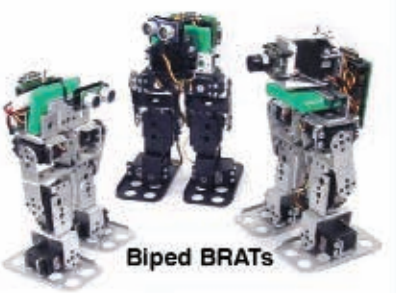
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!

Visit our huge website to see our complete line of robots, electronics, and mechanical components.

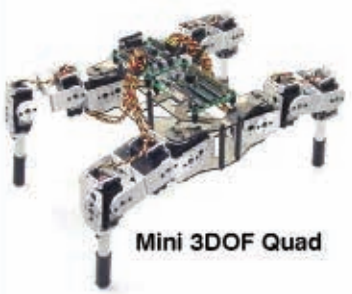
CH3-R



T-Hex



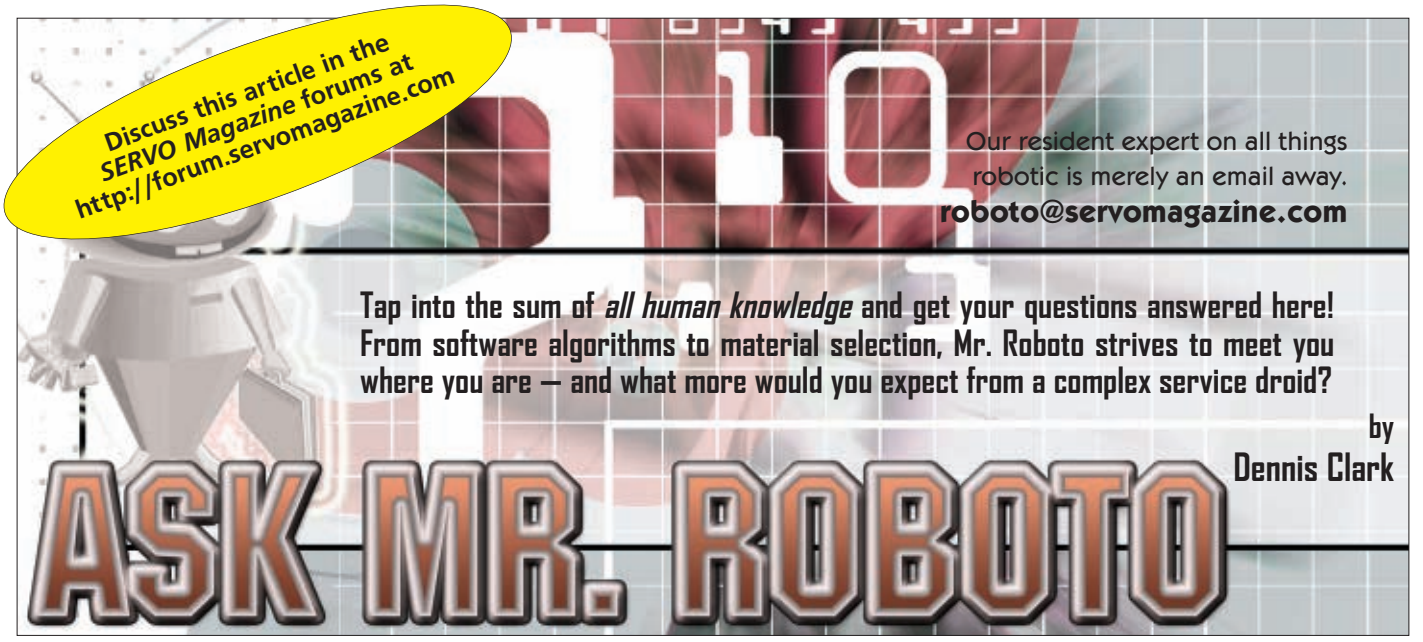
Biped BRATs



Mini 3DOF Quad

Phoenix

Images represent a fraction of what can be made! www.lynxmotion.com The SES now has over 200 unique components!



I got some takers wanting expanded information based on previous columns, so I'll answer one of them in this issue, and add in (as a way of killing two avians with one stone) a short review of the new Microchip MPLABX IDE. As my loyal readers know, I am an Apple kind of guy, so imagine my excitement when I found out a year or so ago that Microchip was coming out with an OS-agnostic IDE for their PIC line! (It made me shiver all right.) I'm kind of fond of Microchip in general, and the higher-end PICs specifically, so except for some robot programs that only run on Windows (hint, hint), this was the last thing I needed to stay in my favorite environment.

MPLABX runs on Netbeans which is a kind of Java developer framework rather like Eclipse. There are releases for MPLABX on Windows, Linux, and OS X, so there is something here for everyone. There are installs for hobbyists who use C compilers for all of the PIC line as well, and they all run on all of the OS platforms. A shiny new toy, so ONWARD!

However ... not a question exactly, but leading up to an answer ...

Before I get into a reader's query, I'm going to briefly show how to set up a project with MPLABX. This will be a bit of a quick start — rather than using a full blown manual — because I firmly believe that any program that requires a manual to use has already failed in its purpose! I'll point you a bit in the right direction, and you can poke around on your own in the IDE's corners and under its covers to discover all that it will do.

The First Step: Install MPLABX v.1.0

To get the IDE and C compilers, go here: **http://ww1.microchip.com/downloads/mplab/X_Beta/index.html.**

On this page, you can select the Platform (upper left of the page) and all of the compilers and documents that you would like. I just clicked on all of the items and hit the *Download Now* button. Remember, I'm an OS X user so my description of this process will be distinctly Apple OS X oriented. When you have gotten through all of the dialogs questioning your sanity in downloading things from a website, you (on a Mac) will have a DMG (disk image) for the IDE, C18, C30, and C32 compilers, as well as a zip file of release notes. If you so chose, you will also have two PICC Hi-Tech ".run" files for the Hi-Tech compilers for the PIC16 and PIC18 devices. The DMGs are obvious; mount them and run the installer packages. Do the IDE first and the compilers will automagically install in places that the IDE will find them. I read on the MPLABX forums that to install the Hi-Tech compilers, you will have to open a terminal window (command line interface) and "cd" to your install location (I used the defaults), and manually run these installs as root:

```
chmod +x picc<blahblah>.run
sudo ./picc<blahblah>.run
```

Remember, OS X is basically UNIX under the hood; these are UNIX commands to run shell scripts that need root access. I've not done this yet, since I don't know how to use the Hi-Tech C compiler yet. After all of this, you have everything installed. Now that you have the IDE installed, plug in your ICD3, PicKit3, Real ICE, or whatever Starter kit or supported programmer/ICD you have for your PIC. The IDE will figure it out. I found that OS X didn't have any problem with my ICD3, so cool.

Creating a Project in MPLABX

I started out by clicking on the tab on the MPLABX "Start Page" called "Import MPLAB Legacy Project." I am going to do this column using the Microchip Explorer 16 Developer Board with the PIC24FJ64GA004 PIM installed, so I imported the *Explorer 16 Demo* program project by just following what the Wizard told me to do. The project

imported with one hitch; I had to hunt down an ".h" file that resided elsewhere in the Microchip projects folder and move it into the new project folder, then change the .c file in which that .h file was referenced. I then built and downloaded to the Explorer 16 and everything ran just fine.

However, we're going to do a *new* project, so a slightly different path is taken. On the *Start Page,* click on the *Create New Project* link; you can also click on the icon in the upper left icon bar that does the same thing. Refer to **Figure 1**.

From here, you'll get the first page of the new project wizard. I've not yet learned all the nuances of this process, so I am sticking with the wizard for now. Pay attention to all of the details on these pages or you'll end up going back and futzing with project properties later on. (Yes, *futzing* is the technical term for "wasting time fixing what I or someone else didn't do right the first time.") Anyway, after you get the project wizard going, you'll see **Figure 2** — the *Choose Project Type* page. Select the *Standalone Project* as **Figure 2** shows.

Click *Next* after choosing. You will now see the *Select Device* page. Select *16-bit MCUs (PIC24)* in the *Family* list, then click in the *Device List* and start typing **PIC24FJ64GA004** until that is what is selected. That is the device I'm going to use to answer the reader's question. See **Figure 3** for the details of my description. Click *Next* again. The next page that will come up will ask you about a header to use with the chip that liberates some I/O pins during debugging. Header? We don't need no stinking header! Ahem, well, I don't have one anyway, so I didn't select the proffered device. I just clicked *Next* and passed this page.

Now, we will select our programmer or ICD (In-Circuit Debugger) that we will use. I have a Microchip ICD3 "Hockey Puck" which I had already plugged in. It showed up in the list under *ICD3* with its serial number. You must select the tool by selecting the specific device you will be using. MPLABX allows you to run multiple debugging sessions with multiple debuggers at the same time, so this associates the device with this project explicitly. If you change devices later on, you can change this association by right-clicking on the project and selecting the *properties* under the project. The green tools are fully supported; the yellow tools are partially supported; so, you can guess what the red dots next to a tool means. MPLAB is still BETA, so not everything is fully supported yet. If you have a third party tool, hopefully it emulates one of the supported

Figure 1. MPLABX Start page.

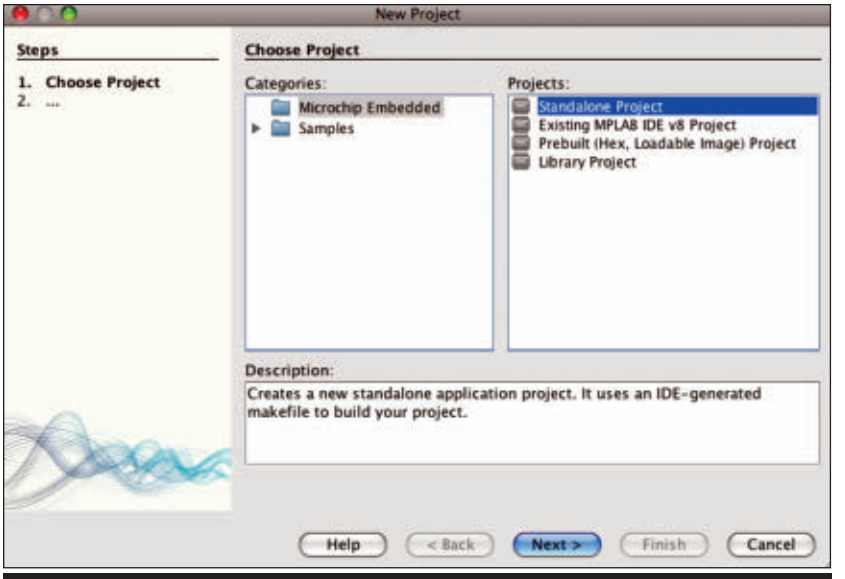


Figure 2. Choose Project page.

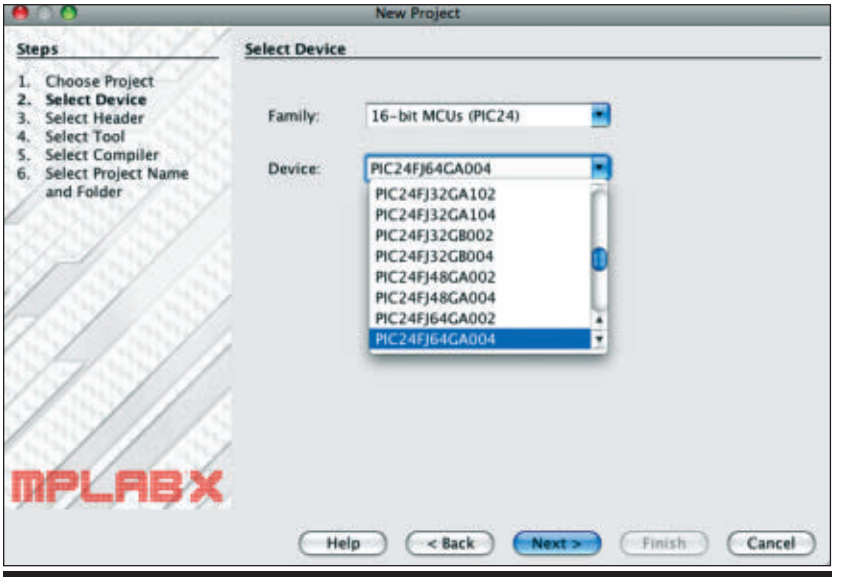


Figure 3. Select Device page.

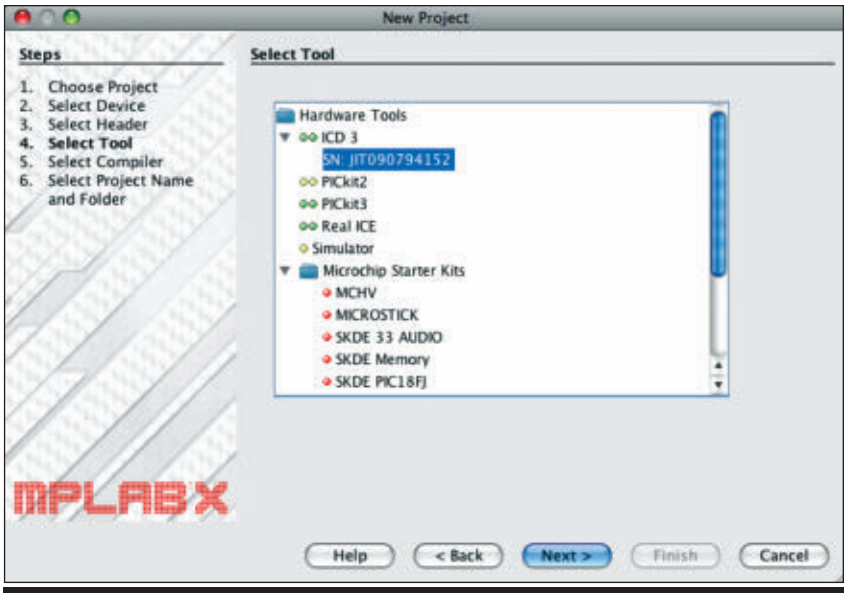


Figure 4. Select Tool page.

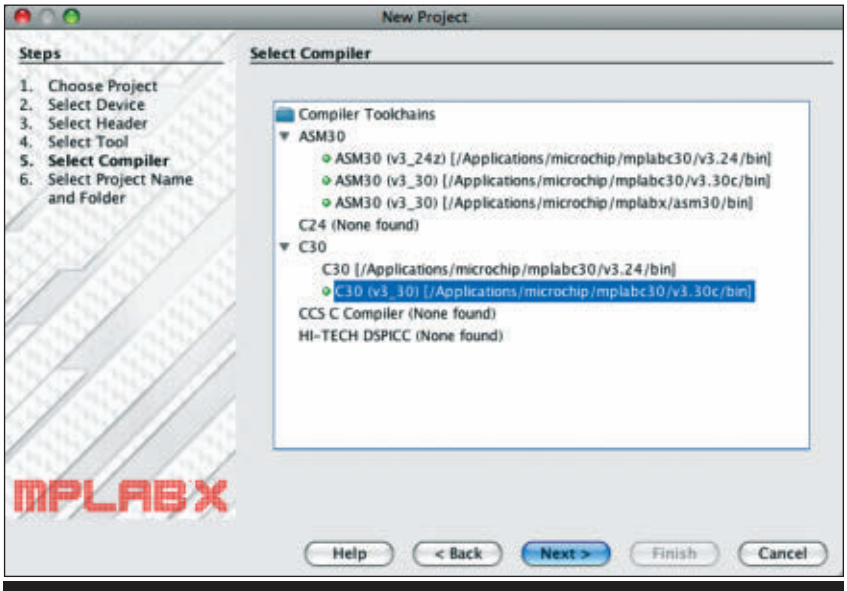


Figure 5. Select Compiler page.

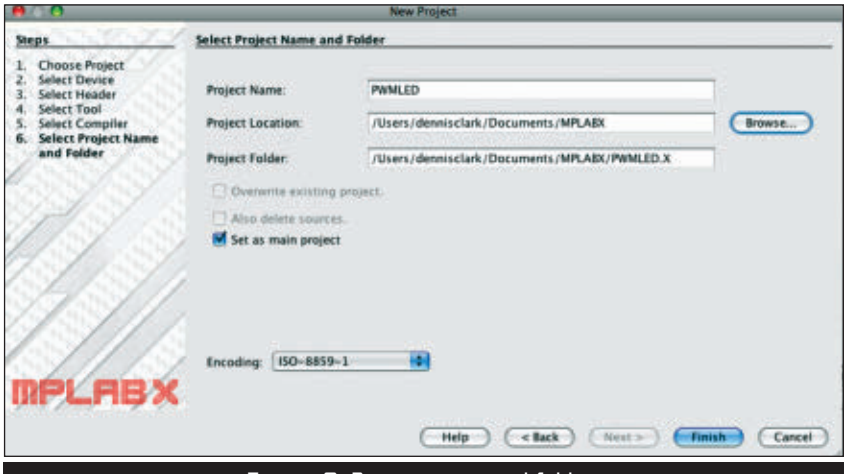


Figure 6. Project name and folder.

Microchip tools well enough to be seen. **Figure 4** shows this page of the wizard. As before, click on *Next* when you're done here.

Now, we select the compiler. When I downloaded, I got the C30 v3.30 compiler. This handles both the PIC24 and the dsPIC parts. The IDE recognized the PIC for the project and pre-selected the correct compiler. MPLAB allows multiple compiler revisions to be available at the same time for selection. This is why you also see the 3.24 C30 compiler. I got that with an earlier BETA version of the IDE. Click *Next* when you are done here (see **Figure 5**).

The last page of the wizard (**Figure 6**) selects a project name and location. I like putting my projects into the *Documents* folder; in this case, I called the folder **MPLABX** and the project **PWMLED** because that is what I'm going to do with the example. After you've filled things in as I did in **Figure 6**, click on *Finish* and we'll move on to add files to the project.

There are a few ways to add files to a project; I'll leave some of them as an exercise for the reader. The two that I used were to drag some source files into the folder created by the wizard and then right-click on the project name in the *Projects* list on the left side of the IDE, and choose "*Add Existing Item*" from the menu that appeared. The other way was to select *New* and add a *C Source File* which can be an ".h" or a ".c" file. I found that the ".c" was easy to select, but I had to manually type the ".h" into the *extension* select list to get the correct file type. **Figure 7** shows what I'm talking about. You can also just click on the white page with the '+' on it in the upper left icon bar to get the file add screen.

To build, you click on the hammer icon shown in **Figure 7**; the hammer with a broom is "clean and build" and the icon with the green arrow from a screen to a chip is — you guessed it — build and download to the chip.

Okay, now we're ready to answer the question and create a specific project!

Q. I have really enjoyed your columns, particularly the ones on timers and the prior ones on FSM. I would be very interested in a column describing setting up the PIC24F OCP/PWM. I have been using PIC16Fs as well as the Arduino for a while, and am going to take your advice and step-up to the 24F series for my robots.

— Steve Ghertner

A. Thanks for reading and asking questions! I'll be happy to give an example of PWM on the PIC24 line. Compared to the older (SO 20th century) PIC, doing PWM is simple on the PIC24FJ line, with one exception: the *Peripheral Pin Select* operations. However, with a little effort, even that gets simpler. **Figure 8** shows the Microchip Explorer 16 board that I am using for this example. It has buttons, LEDs, a temperature sensor, voltage sensor (on a pot), LCD display, PICTail interface, and other bits and pieces of useful stuff to experiment with. **Figure 8** also shows the ICD3 programmer/debugger that I use for most of my hobby work. We'll select one of the LEDs on the board to dim and brighten using PWM on a configurable I/O output line of the PIC24FJ64GA004.

The most useful and confusing part about the PIC24 chips are the configurable input and outputs that can be assigned to hardware blocks. The I^2C interface and the analog I/O can't be changed around; there are two or three programming ports you can select for use but they are separate blocks, not configured I/O lines to a single hardware function. The rest of the special hardware blocks can be assigned to any available I/O line that isn't fixed. Hardware blocks like the USART, SPI, or PWM fit into this category. This is kind of useful if you build a board and then change your mind about where you are putting a particular function.

For this example, I have chosen the I/O line for **LED3** as the Explorer 16 dev board defines it. This LED is on the I/O line *LATB, PIN9*. Or, IOPortB latch, bit 9. The PIC24 registers are all 16 bits wide, except for those that are 32 bits wide. This includes the configuration registers, as well as the I/O ports and latches, and timer registers. In

TABLE 10-3: SELECTABLE OUTPUT SOURCES (MAPS FUNCTION TO OUTPUT)

Function	Output Function Number(1)	Output Name
NULL(2)	0	NULL
C1OUT	1	Comparator 1 Output
C2OUT	2	Comparator 2 Output
U1TX	3	UART1 Transmit
$\overline{U1RTS}$(3)	4	UART1 Request To Send
U2TX	5	UART2 Transmit
$\overline{U2RTS}$(3)	6	UART2 Request To Send
SDO1	7	SPI1 Data Output
SCK1OUT	8	SPI1 Clock Output
SS1OUT	9	SPI1 Slave Select Output
SDO2	10	SPI2 Data Output
SCK2OUT	11	SPI2 Clock Output
SS2OUT	12	SPI2 Slave Select Output
OC1	18	Output Compare 1
OC2	19	Output Compare 2
OC3	20	Output Compare 3
OC4	21	Output Compare 4
OC5	22	Output Compare 5

Figure 9. Configurable output pins.

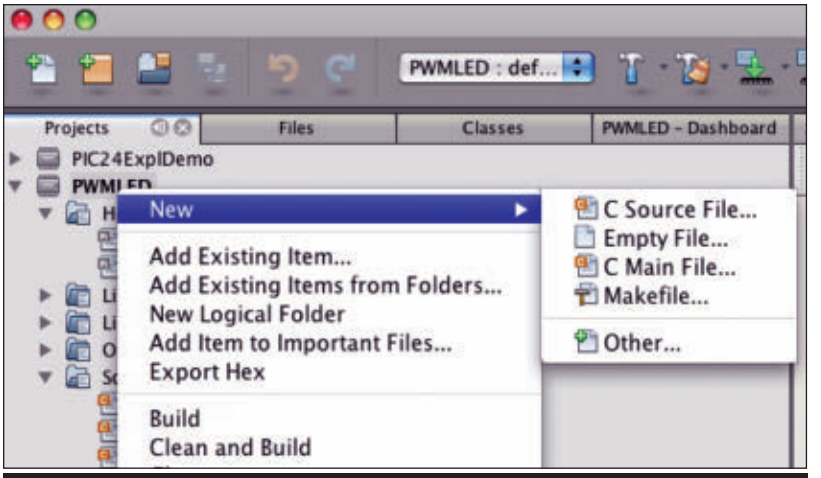


Figure 7. Adding files.

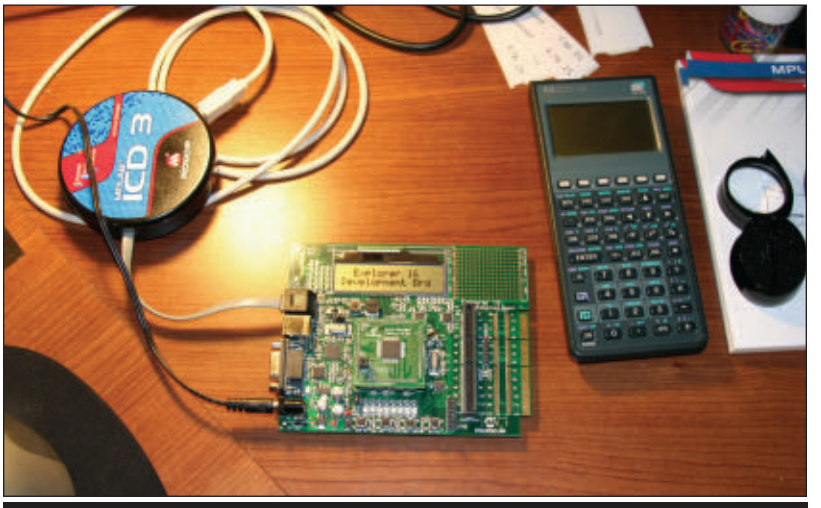


Figure 8. The Microchip Explorer 16 dev board.

REGISTER 10-19: RPOR4: PERIPHERAL PIN SELECT OUTPUT REGISTER 4

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP9R4	RP9R3	RP9R2	RP9R1	RP9R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP8R4	RP8R3	RP8R2	RP8R1	RP8R0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-13 **Unimplemented:** Read as '0'

bit 12-8 **RP9R4:RP9R0:** Peripheral Output Function is Assigned to RP9 Output Pin bits (see Table 10-3 for peripheral function numbers)

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **RP8R4:RP8R0:** Peripheral Output Function is Assigned to RP8 Output Pin bits (see Table 10-3 for peripheral function numbers)

Figure 10. PPS register for RP9.

Listing 1: I/O locking and setting functions.

```
void ioMap(void)
{
//(__PIC24FJ64GA004__)
    //INPUTS **********************

    //U2RX = RP19
    RPINR19bits.U2RXR = 19;
    //SDI2 = RP20
    RPINR22bits.SDI2R = 20;

    //OUTPUTS *********************

    //RP25 = U2TX
    RPOR12bits.RP25R = U2TX_IO;
    //RP24 = SCK2
    RPOR12bits.RP24R = SCK2OUT_IO;
    //RP21 = SDO2
    RPOR10bits.RP21R = SDO2_IO;

    //RP9 = RP9 = LED3
    RPOR4bits.RP9R = OC3_IO;

}

void lockIO(void)
{
    asm volatile ("mov #OSCCON,w1 \n"
                "mov #0x46, w2 \n"
                "mov #0x57, w3 \n"
                "mov.b w2,[w1] \n"
                "mov.b w3,[w1] \n"
                "bset OSCCON, #6");
}

void unlockIO(void)
{
    asm volatile ("mov #OSCCON,w1 \n"
                "mov #0x46, w2 \n"
                "mov #0x57, w3 \n"
                "mov.b w2,[w1] \n"
                "mov.b w3,[w1] \n"
                "bclr OSCCON, #6");
}
```

Listing 2: Output mapping of hardware functions.

```
//PPS Outputs
#define NULL_IO             0
#define C1OUT_IO            1
#define C2OUT_IO            2
#define U1TX_IO             3
#define U1RTS_IO            4
#define U2TX_IO             5
#define U2RTS_IO            6
#define SDO1_IO             7
#define SCK1OUT_IO          8
#define SS1OUT_IO           9
#define SDO2_IO             10
#define SCK2OUT_IO          11
#define SS2OUT_IO           12
#define OC1_IO              18
#define OC2_IO              19
#define OC3_IO              20
#define OC4_IO              21
#define OC5_IO              22
```

the time honored fashion of programmers everywhere, I copied the Microchip programmer's habit of putting the definitions and setups of the configurable hardware in the *iomapping.h* and *iomapping.c* files of my project. As you will see later, this makes dealing with the Peripheral Pin Select mapping a little more obvious. Here is the thought process:

The documentation for the Explorer 16 dev board and the PIC24FJ64G004 PIM (plug-in module) says that LED3 is on port RB9. The pin-out for the device shows us that RB9 is RP9 which is the reprogrammable pin 9.

In section 10.4 of the PIC24FJ64GA004 family spec sheet. (Table 10-3), we see that OC3 (PWM3) is Output Function 20. Look at **Figure 9**.

Now, we know where the LED is and what the OC3 hardware block is. We can now assign the PWM to that I/O line. Section 10.5 of the document has all of the registers that assign the hardware to an I/O line. The Pin Select Input register has five-bit values you can set to select which RPn line you wish to use as an input for the register's hardware block input line. There are two hardware block selection values for each register.

When selecting an output line for a hardware block as we are, it is done differently. The register has a five-bit number that you enter that defines the hardware block (OC3 is 20) you are assigning to the RPn pin that value represents. Did you get that? For an input, you select the I/O line you want to assign to the hardware block the register represents. For an output, you select the hardware block you want the RPn output pin to be connected to. **Figure 10** shows the register we use to attach OC3 to RP9 which is also RB9, which the LED3 is connected to.

Before you can set these bits though, you need to unlock the PPS hardware. This is where the *lockIO(), ioMap(),* and *unlockIO* functions in **iomapping.c** come into play. See **Listing 1**.

Note that I set OC3_IO in the ioMap() function. I got this assign from the **iomapping.h** file in **Listing 2**. This is a convenient way to document these instead of "magic numbers" that you just code into your source.

Finally, we get to actual code! **Listing 3** shows how to configure the OC3 (PWM) output and then dim and brighten LED3 on the Explorer 16 dev board. I chose the PWM period just to have a fairly low frequency so that the LED would be brighter; it may not be the perfect frequency, so feel free to experiment. Notice that the PWM initialization is simple. Then, all you need to do is set the OC3RS register to a value that is less than or equal to your period (PER_MAX) to change the pulse duty cycle. Your resolution will depend on your period clearly; a faster period will leave you with lower resolution. Our choice of 32768 gives you 15 bits of PWM resolution. I've dimmed an LED here, but connect this output to your favorite DC motor driver and you have motor speed control whose resolution will depend on your PWM frequency choice.

There you have it! More information on the care and feeding of the PIC24F line of 16-bit microcontrollers. PWM

Listing 3: Setting up the PWM and lighting the LED.

```
#define PER_MAX 32768          // about 122Hz

void InitPWM(void)
{
     OC3CONbits.OCM = 6;       // Output continuous pulses
     OC3CONbits.OCTSEL = 1;    // TIMER3 is clock source
     T3CON = 0x8000;           // prescale 1, internal clock,
turned on
     PR3 = PER_MAX;            // Currently about 122Hz
}

int main(void)
{
     unsigned long delay;

     /*
      * Set up the hardware PPS functionality
      */
     unlockIO();
     ioMap();
     lockIO();

     // Setup the timers and PWM
     Timer2Init();
     InitPWM();

     // Make the LED pin an output
     LED3_TRIS = 0;

     // Show what full on brightness is
     OC3RS = PER_MAX;
     delay = t_1ms + 2000;
     while(delay > t_1ms);

     // Start LED off
     OC3RS = 0;
     /*
      * This loop will cycle the LED from off to
      * full brightness, then off and start over.
      * Change the delay to step faster or slower,
      * change the increment to change the step
      * size.
      */
     while(1) {
          delay = t_1ms + 70;
          while(delay > t_1ms);
          OC3RS = OC3RS+500;
          if (OC3RS > PER_MAX) {
               OC3RS = 0;
          }
     }
```

is simple with these parts, even with the need to use PPS to set the hardware output lines. Keep this in mind though: You can unlock your hardware and change those PWM output lines programmatically. This means that you can change what signals you send where dynamically if you need to. That is useful flexibility that makes the added complexity worth while.

The source code for this project is available in the article downloads as PWMLED.zip.

Well, that wraps up a long discussion for a single question, but sometimes it helps to be shown how to take the next step. I am working up to some interesting things in the future, but they take a while to do, so in the meantime keep those questions coming! I love to hear about projects in the works and love to help you get over the humps. Drop me an email at roboto@servomagazine.com. Until next time, keep on building robots! SV

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**.

— R. Steven Rainwater

MARCH

9-10 **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana-Champaign, IL
Teams build autonomous robots that compete in a game that changes each year.
http://jsdc.ec.illinois.edu

10 **Trenton Computer Festival Robotics Contest**
Ewing Township, NJ
Every year is different, but robot events usually include maze navigation, precipice avoidance, and Micromouse.
www.tcf-nj.org

10-11 **RobotChallenge**
Vienna, Austria
Events include Parallel Slalom, Slalom Enhanced, Sumo, Mini Sumo, and Micro Sumo.
www.robotchallenge.org

16-20 **Apogee iStrike**
BITS Pilani KK Birla Goa Campus, Zuarinagar Goa, India
Autonomous ground robots must follow traffic signs while avoiding obstacles.
www.bits-apogee.org

24 **CIRC Central Illinois Bot Brawl**
Lakeview Museum, Peoria, IL
Lots of events including line following, line maze, Sumo, RC combat, and Best of Show.
http://circ.mtco.com

24 **Greater Philadelphia Sea Perch Challenge**
Drexel University, Philadelphia, PA
Tethered underwater ROV event.
www.phillyseaperch.org

24 **Harrisburg University Pennbots**
Harrisburg University, Harrisburg, PA
Maze solving plus remote control vehicle combat.
http://web.me.com/wjbechtel/Robot_Competition/Welcome.html

30 **Trinity College Fire Fighting Home Robot Contest**
Trinity College, Hartford, CT
Contest runs through April 1. Autonomous robots must navigate a mock house, locate a fire, and extinguish the flames.
www.trincoll.edu/events/robot

APRIL

12-14 **National Robotics Challenge**
Marion, OH
This year's event is the Canine Companion Challenge.
www.nationalroboticschallenge.org

14 **Brown IEEE Robotics Competition**
Brown University, Providence, RI
Here, 25 cm autonomous robots must navigate a maze.
http://brown.edu/Departments/Engineering/Organizations/Ieee/competition

14 **RoboRodentia**
California Polytechnic, San Luis Obispo, CA
Autonomous Micromouse-like robots must navigate a maze.
https://sites.google.com/site/calpolycomputerengineering

19-21 **VEX Robotics World Championship**
Anaheim, CA
Best high school and university VEX teams in the world compete.
www.vexrobotics.com/competition

20 **Carnegie Mellon Mobot Races**
CMU, Pittsburgh, PA
Events include Mobot Slalom and the MoboJoust.
www.cs.cmu.edu/~mobot

20-22 **RoboGames**
San Mateo Event Center, San Mateo, CA
Events include FIRA, BEAM, Mindstorms, and machine combat.
www.robogames.net

21 **Penn State Abington Fire-Fighting Robot Contest**
Penn State Abington, Abington, PA
Autonomous robots must navigate a maze and extinguish a fire.
www.ecsel.psu.edu/~avanzato/robots/contests

21 **Robot-SM**
University of Gothenburg, Gothenburg, Sweden
Events include Sumo, Mini Sumo, and Robot Pentathlon. **www.robotsm.se**

21 **The Tech Museum of Innovation's Annual Tech Challenge**
Parkside Hall, San Jose, CA
Shake, Rattle, and Rescue — robotic rescue devices for earthquakes.
http://techchallenge.thetech.org

21 **UC Davis Picnic Day Micromouse Competition**
University of California, Davis Campus, CA
Micromouse maze solving.
www.ece.ucdavis.edu/umouse

23-24 **IEEE TERPA Student Robotics Competition**
Boston, MA
This year's theme is robotic cars.
www.ieeerobot-tepra.org/studentcomp.html

25-28 **FIRST Robotics Competition**
Edward Jones Dome, St. Louis, MO
FIRST teams from all over gather to compete at the annual championship.
www.usfirst.org

NEW PRODUCTS

High Performance Intelligent Inertial Sensors

YEI Technology is using patent-pending technologies to develop their YEI 3-Space Sensor™ family of intelligent Inertial Measurement Units (IMUs)/Attitude and Heading Reference Systems (AHRS). With prices starting around $100, advanced inertial technology is entering into price-sensitive and mass-market applications without sacrificing performance.

Each 3-Space Sensor uses a triaxial gyroscope, accelerometer, and compass sensors in conjunction with advanced onboard filtering and processing algorithms to determine orientation relative to an absolute reference orientation in real time. The product family offers a breadth of communication, performance, and packaging options ranging from the ultra-miniature surface-mountable TSS embedded module to fully integrated battery-powered TSS Bluetooth, TSS wireless 2.4 GHz, and TSS data-logging versions.

Orientation can be returned in absolute terms or relative to a designated reference orientation. The proprietary multi-reference mode increases accuracy and reduces and compensates for sensor error. The 3-Space Sensor system also utilizes a dynamic sensor confidence algorithm that ensures precision across a wide range of operating conditions. Fully filtered orientation (AHRS) outputs at rates up to 200 Hz and sensor data (IMU) outputs at rates up to 1,000 Hz.

The 3-Space Sensor family features are accessible via an open communication protocol that provides access to all available sensor data and configuration parameters using a variety of communication interfaces. Versatile commands allow access to raw sensor data, normalized sensor data, and filtered absolute and relative orientation outputs in multiple formats including: quaternion, Euler angles (pitch/roll/yaw), rotation matrix, axis angle, two vector (forward/up). Customers can use the free 3-Space Suite to easily visualize this data in a graphical format, as well as log data for future analysis.

For further information, please contact:

YEI Technology

Website: **www.3SpaceSensor.com**

Pattern Clamping Hubs

ServoCity's new 0.770" pattern clamping hubs offer an inexpensive way to attach hub mount gears or wheels to a shaft. These clamping hubs offer several advantages over set screw hubs. First, they do not damage the shaft which they are attached to. Second, they offer more holding power in high torque applications. The side clamping machine screw is 6-32 and accepts a 7/64" hex key.

In addition to the new clamping hubs, ServoCity also offers 0.770" pattern set screw hubs. One advantage of set screw hubs is that they are more equally balanced for high speed applications. The hubs offer four equally spaced 6-32 tapped holes and are precision machined from 6061 T6 aluminum. The set-screw is 10-32 and uses a 3/32" hex key. Both the new hubs are available with the following bore sizes: 3 mm - 6 mm and 1/8" - 1/2".

Pan & Tilt System

Also available from ServoCity is a new patented point and shoot PT785-S pan and tilt servo head. This precision, closed-loop system is designed using two HS-785HB servos to manipulate cameras/devices up to 6 lbs – even though the entire

head only weighs 1.8 lbs. Each axis can operate up to 400° of rotation and ABEC-5 ball bearings support the hollow aluminum shafts that allow video and power wires to be routed through the points of rotation.

The PT785-S is perfect for still photography, amateur videography, or security applications. It can easily be mounted to various jib-cranes in the hanging or upright position using the supplied 90 mm diameter base. Balancing your camera or device is simple, using the vertical slide adjustment. The system is supplied with a 4:1 gear ratio, but optional gear ratios are available (which will extend or limit total rotational travel). A fully assembled system is $349.99.

For more information please contact:

ServoCity

Website: **www.servocity.com**

Sport Servos

Hitec is now offering the high voltage digital HS-5565MH and HS-5585MH sport servos. These premium sport servos feature coreless motors, heavy duty metal gears, and dual ball bearings. Rated at 7.4 volts for 2S LiPo operation, they provide durability and power for RC hobby demands.

The HS-5565MH brings speed and acceleration to 1/10th scale touring cars, buggies, and short course vehicles; helicopters up to .90 size and sport to high performance aircraft up to 25 percent. The HS-5585MH supplies 1/10th monster trucks to 1/8th scale buggies and truggies, and high performance airplanes up to 35 percent with outstanding strength and precision. The estimated retail price is $69.99.

Four-Port Multicharger

Hitec is also offering another economical battery charger called the X4-Eighty.

Featuring four identical and independent 80 watt power outputs, this easy-to-operate, microprocessor-controlled charger has a total output power of 320 watts. Each port is capable of charging all types of rechargeable batteries at up to six amps, whether it is a 6S lithium pack to 15 NiCd/NiMH cells, to 6-12 volt lead acid batteries. Four individual balancing ports eliminate the need for a separate balancer when charging lithium batteries.

Equipped with a twin fan cooling system and an internal sensor for controlling fan speed, the X4-Eighty delivers the reliability and safety required in charging various types of batteries. Its powerful capability combined with its reliable safety functions make it a good choice for electric and nitro modelers. The estimated retail price is $169.99.

For further information, please contact:

Hitec

Website: **www.hitecrcd.com**

Bi-Polar Chopper Stepper Motor Drive Kit

Global Specialties new GSK-187 bipolar chopper stepper motor drive kit will drive bipolar stepper motors up to 2A per phase with a 9-35V supply. The kit features full and half stepping. All signals (step, direction, enable) can be interfaced to external +5V logic or a microcontroller. Components and an instruction manual included.

Global Specialties has added six new kits, expanding their line to 17 kits. Each kit comes complete with detailed assembly instructions and all necessary parts.

Other new kits include the Temperature Meter Kit, Basic Power Supply Kit, Fiber Optic Audio Link Kit, 3-1/2 Digit Panel Meter Kit, and the Step Up DC Converter Kit. For further information, please contact:

Global Specialties, LLC

22820 Savi Ranch Parkway
Yorba Linda, CA 92887
Tel: **800 • 572 • 1028**
Website: **www.globalspecialties.com**

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

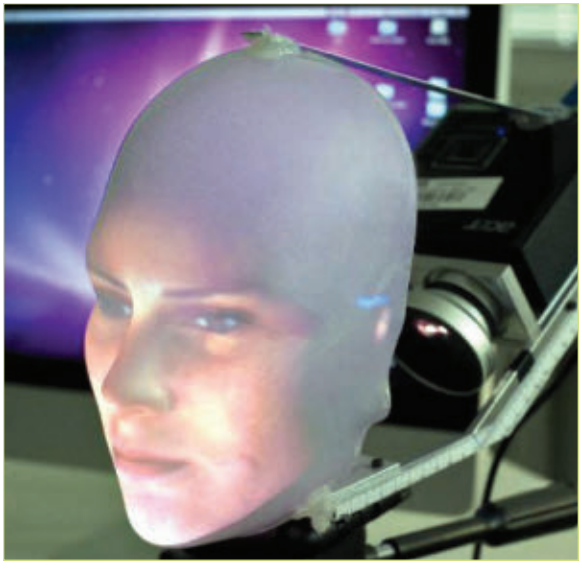
VIVA LA FRANCE'S DRIVING ROBOT

Autonomous vehicle projects are picking up speed — literally. One project in particular is an interesting self-driving vehicle developed in France. (See photos on next page.)

The car was engineered by a team from IFSTTAR (a French R&D organization) and the Embedded Electronic Systems Research Institute at ESIGELEC (an engineering school in Rouen, in the Normandy region). The goal is to develop autonomous vehicle technologies that can help test automotive safety systems. By using a "driving robot," the researchers can control the exact trajectory, speed, and behavior of a vehicle. "Then we can compare the performance of different safety systems," says Pierre Merriaux, one of researchers involved.

The group modified a Renault Grand Espace by adding a Stahle robot driver, sensors, cameras, and a control bay on the roof. Merriaux explains that the car is guided by GPS RTK and an iXSea inertial unit, with data acquired and processed using the RTMaps multisensor engine. There are three cameras to monitor the vehicle's surroundings and one forward-facing to track road lanes and markings. A LIDAR unit at the front detects other cars and pedestrians. The researchers first tested their robot car manually, driving it with a joystick. Then, they let the car drive itself on a test track. They plan to use the vehicle to evaluate safety features under various driving conditions.

The project is part of a large R&D program called Quasper which involves a number of French companies and labs, including the Thales Group and INRIA. The aim is to develop sensors and "perception systems" for applications in transportation and security.



(Credit: Uli Benz / Technische Universitaet Muenchen)

LET'S FACE IT

Robotics researchers in Munich have joined forces with Japanese scientists at AIST (the National Institute of Advanced Industrial Science and Technology) to develop an ingenious technical solution that gives robots a human face. By using a projector to beam the 3D image of a face onto the back of a plastic mask and a computer to control voice and facial expressions, the researchers have succeeded in creating Mask-bot — a startlingly human-like plastic head. Before this technology is used to give robots of the future a human face, it may well soon be used to create avatars for participants in video conferences.

Mask-bot can already reproduce simple dialog. When Dr. Takaaki Kuratate says "rainbow" for example, Mask-bot flutters its eyelids and responds with an astoundingly elaborate sentence on the subject: "When the sunlight strikes raindrops in the air, they act like a prism and form a rainbow." When Mask-bot talks, it also moves its head a little and raises its eyebrows to create a knowledgeable impression.

"Mask-bot will influence the way in which we humans communicate with robots in the future," predicts Prof. Gordon Cheng, head of the Institute for Cognitive Systems (ICS) at TU München team. The researchers developed several innovations in the course of creating Mask-bot.

Although other groups have developed three-dimensional heads, these display a more cartoon-like style. Mask-bot, however, can display realistic 3D heads on a transparent plastic mask, and can change the face on-demand. A projector positioned behind the mask accurately beams a human face onto the back of the mask, creating very realistic features that can be seen from various angles, including the side.

WE'VE GOT THE BEAT

Robotics and high tech toys manufacturer, TOSY Robotics JSC, has launched an advanced portable speaker that also functions as a dancing robot. Pop and R&B singer Justin Bieber appeared with the brand new robot at the recent Consumer Electronics Show (CES) at the TOSY Robotics booth.

The mRobo is operated with a battery and features 2 GB of internal memory. Users can upload approximately 500 songs into the device with the help of a USB port; they can choose their desired tracks by using a remote. The music will play via a speaker and can generate bass at 40 Hz. The portable speaker can listen to music from other devices, to its own music, or music flowing through Bluetooth. Once the music starts playing, the portable speaker will immediately change — growing a head, arms, and legs — and will dance via a range of pre-programmed moves to almost any type of music.

TOSY Robotics JSC's Founder and CEO, Vinh Hoang stated that the mRobo is an innovative entertainment device because it enables music lovers to enjoy their passions in an innovative way. The device introduces a novel way to listen to their favorite songs and also provides a new partner to dance to their desired music.

The mRobo capability to change back into a speaker makes it portable and suitable for today's generation. It measures 4.3"W x 6.3"L x 7.9"H, and weighs around 3.3 lbs. Its height is less than 18" when it changes into a robot. The mRobo utilizes software to study the music for rhythms and beats which it then responds to with matching moves.

SCENT OF A ROBOT

Want to know what's missing in your life? Yes, that's right! A robot that wanders around your house seeking out bad smells and neutralizing them. Obviously, there's a huge market for these things because there were two of them at the Consumer Electronics Show.

First up is Moneual's Rydis H800 — a rather gigantic mobile air purifier. Inside are six air purification systems, including a washable pre-filter, a functional filter, a HEPA filter, an activated carbon filter, a semi-ULPA filter, and an impregnated activated carbon filter. The robot can run for four hours on a charge, autonomously navigating around with what looks to be a suite of ultrasonic sensors. Or, you can kick it up to TURBO MODE which may just halve the battery life.

If you are really concerned with all those nasty smells that hide up near the ceiling, look no further than the Ecovacs A330. It has a "unique HACM chemically absorbing system" that "breaks down toxic gasses such as formaldehyde" which is great if, you happen to have a whole bunch of formaldehyde floating around. The robot can extend upwards by a foot or so (about 30 cm) which must help it reach higher smells.

NICE KITTY

Taylor Veltrop's Nao teleoperation saga has been going on for about a year now, since Veltrop — a Japan-based roboticist — had a robot doing push-ups for him. However, there's more to teleoperation than just virtual exercise. Yep. There's also virtual pet pampering and it involves Nao, a Kinect system, a treadmill, a bunch of cameras, and one exceptionally tolerant cat. There's actually a lot going on here. The system consists of a Nao (of course), a Kinect sensor, two Wiimotes, a treadmill, a camera mounted on Nao's head, and a head-mounted immersive display system for the user that also controls Nao's head and neck. So, put this all together and Veltrop (the user) can walk forward and Nao walks forward. Veltrop can turn and Nao turns. Veltrop can pick up a brush and start brushing on a clearly uncomfortable cat, and there you have it. This is a fairly remarkable feat of robotics, and it's worth noting just how much of this was accomplished based on off-the-shelf (and relatively inexpensive) sensors combined with some intense cleverness. In just the last few years, the availability of cheap and open source software and hardware has enabled people without giant research budgets to do some amazing things that just haven't been possible before.

BOT-TENDERS

Beef up your bar with these robotty Swizzle Sticks made of acrylic with ceramic heads. The set of four comes in different styles and are each 9" long. For those who prefer wine, the double lever Corkscrew has a standard spiral with cap lifter and is made of stainless steel. You can find them at **amazon.com**.

USING YOUR BLOCK HEADS

Now, you can build a robot bit by bit with Cubelets from Modular Robotics that snap together automatically, and can move, emit noise and light, and detect temperature and motion. Each Cubelet uses logic to determine what the other blocks will do without any additional programming, and is a smaller robot in itself. For example, if you set two speaking blocks and a battery block together they will communicate. Modular Robotics is a new company which is a spin-off from Carnegie Mellon University. They are turning their research prototype construction toy into a commercial product for science centers, children's museums, and hobbyists. They believe that toys shape the way that children think about the world. The Cubelets KT01 standard kit comes with 20 magnetic blocks that can be snapped together to make an endless variety of robots with no programming and no wires. You can build robots that drive around on a tabletop, respond to light, objects, and temperature, and have surprisingly lifelike behavior. Instead of programming that behavior, you snap the Cubelets together and watch the behavior emerge as with a flock of birds or a swarm of bees. Each Cubelet in the kit has different equipment on board and a different default behavior. There are Sense blocks that act like eyes and ears; they can sense light, temperature, and how far they are away from other objects. Just like with people, the senses are the inputs to the system.

On the flip side, the Action blocks act as outputs. They do things. Some have little motors inside of them so they can drive around or spin one of their faces. There are blocks that make noise, shine a flashlight, or display their information through a light-up bar graph. Each Cubelet has a tiny computer inside of it and is a robot in its own right. So when you put blocks together, you're actually making a robot out of several smaller robots. Each block communicates with its neighbors, so you know that if two blocks are next to each other, they're talking. If you make a simple robot by connecting a Brightness Sense block to a Speaker Action block, they'll start to talk; when the light in the room gets brighter, the Speaker will get louder. (Actually, you'd need a third block to make this work; every robot needs a Battery block to run.) Next, you could swap the Speaker for a Drive block, and when the light gets brighter, the robot will drive faster. A third category of blocks is the Think blocks. Maybe you'd want to put an Inverse block in between the Brightness and Drive blocks. Then, the robot would drive slower as the light gets brighter. This simple communication between adjacent blocks is what gives the kit a little bit of magic.

Cubelets are designed for people age eight and older.

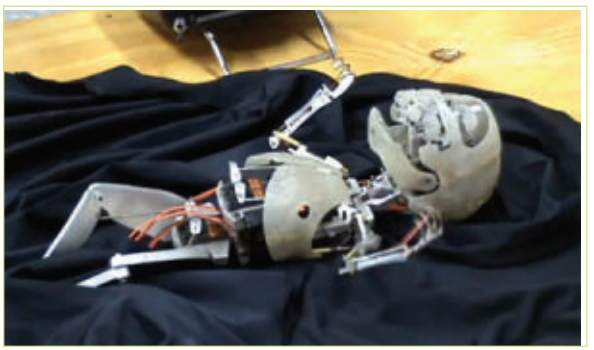
EYE3 SPY

It's actually fairly easy to buy your own camera-equipped flying robot these days. However, stepping up to something a little bit more advanced (say, with an autopilot and enough payload for a serious camera) is both intimidating and expensive. A project covered on Kickstarter — the popular website that crowdsources funding — aims to take all of the stress out of buying and using a professional flying camera platform, while saving you a pile of money (perhaps) at the same time.

The eye3 drone is more or less everything you could possibly want in a robotic aerial camera platform. It's a hexacopter with some serious chops: Each of its six motors puts out four pounds of thrust, giving the eye3 (as a whole) nearly three horsepower and 24 pounds (about 11 kilograms) of vertical thrust. Since the eye3 itself only weighs about five pounds, that gives you a huge amount of payload. Or, another way of looking at it is that most of the time, the motors won't have to be working very hard, which will extend their lifespans and make the platform more reliable. And if all else fails, the eye3 can land itself with just four of its six motors operating.

Having a powerful and capable platform is one thing, but unless you've been flying R/C helicopters or quadrotors for years, getting this platform to do what you want without crashing it is very difficult. This is why eye3 comes with a pre-configured version of the open source AM2 autopilot which uses an inertial measurement unit (or IMU), a GPS, a barometer, a magnetometer, and even optional sonar to manage takeoffs, stable flight, waypoint navigation, and (most importantly) landings without needing you around to screw things up.

The vehicle itself will run you about US $1,000. The autopilot (hardware and software) adds another $500 to that. For a total of $2,500, you also get a transmitter and receiver (and some extra spares and batteries and stuff). It may sound like a lot, but if you take a look at comparable ready-to-fly platforms with autopilots and heavy lift capacity, you're potentially saving yourself thousands of dollars by going with this project.



BABY ONLY A BOT COULD LOVE

This is the Uncanny Valley at its finest. People shouldn't make robot babies. Okay. It's a little bit more excusable if you're a professional animator and someone is paying you to make one of these, um ... things ... but that doesn't make it any less unsettling. In this case, the creepy part isn't so much that this robot doesn't look like a baby. It's the fact that it doesn't look like a baby while simultaneously acting very much like a baby. Getting a robot to act so pseudo-convincingly is largely due to the skills of the baby's, um, driver (?), who appears to be an English professional animatronic creature designer named Chris Clarke. Apparently, Chris has a lot of experience making animatronic robots for movies and TV.

BLUE PLATE SPECIAL

Professor Peter Corke. (Image courtesy of Queensland University of Technology.)

A flying robot as small as a dinner plate that can zoom to hard-to-reach places, and a fleet of eco-friendly robotic farm-hands are just two of the projects a robotics team at the Queensland University of Technology (QUT) is working on.

The pint-sized propellor powered robots can be packed away in a suitcase. They have multiple cameras which enable them to 'see' the world around them as they navigate their way through buildings, carrying out tasks like deliveries or inspections.

"You'll be able to put your suitcase on the ground, open it up, and send the flying robot off to do its job," said Professor Peter Corke, from the Faculty of Built Environment and Engineering.

"These robots could fly around and deliver objects to people inside buildings and inspect things that are too high or difficult for a human to reach easily.

"Instead of having to lower someone down on a rope to a window on the seventh floor or raise them up on a cherrypicker, you could send up the flying robot instead."

The QUT researchers are using cost-effective technology, so the robots are affordable. Within the next year, it may be possible to attach arms to the device so it can also fix things.

DUST IN THE BIN

If you own a robotic vacuum cleaner, you probably know that one of the most annoying things about these devices is that when their dust bin is full, *you* have to clean your cleaning robot. However, some good news from CES is that there are a couple of robot vacuums from Asia that come with docking stations capable of emptying the dust bins automatically.

These are not the first robot vacuums to have the self-emptying dust bin feature, though. There's one from Germany that costs US $1,300 that can do this, and iRobot has had a patent on the books for a few years now, though the company hasn't made any announcements.

The first vacuum comes from a company that you've probably never heard of called Ecovacs. They're based out of China and their Deebot D76 sports the following tagline: "It's a vacuum cleaner, it's a robot, and it's anything you want it to be! Use Deebot D76 as anything you like!"

Deebot moves around using "28 radar navigators" which probably means that it doesn't make active maps. It cleans, avoids obstacles, doesn't fall off of things, etc., but the interesting bit is the charging dock with an integrated vacuum system that both sucks the dirt out of the robot's vacuum bin and doubles as a portable vacuum cleaner.

The second vacuum comes from a company that you've definitely heard of: Samsung. The NaviBot-S includes an "auto dust emptying system," and when the robot senses that its dust bin is full, it heads back to the dock to get its dust sucked out. As a plus, it gets its brushes cleaned to boot. Since the NaviBot makes an active map as it goes (using a camera pointed at the ceiling along with infrared sensors), it can then head back to right where it left off to finish cleaning a room.

Supposedly, you'll be able to buy these vacuums in the US within a year or two, but you hear this type of thing nearly every single year at CES and nothing ever happens. It does seem likely that Samsung's Navibot will show up in Australia, but contrary to what's been heard from the Samsung reps, a US launch isn't necessarily going to follow (one issue is iRobot's intimidating arsenal of robot vacuum patents). As for the Deebot, well, that's even less certain.

DOG DAY AFTERNOONS

Many animals — and lots of humans — seem to have an instinctual aversion to snakes. Many animals (and humans) also seem to have an instinctual aversion to robots. Couple that with an instinctual aversion to running around in disaster zones, and it's remarkable that this robosnake-deploying disaster dog even shows up for work in the mornings.

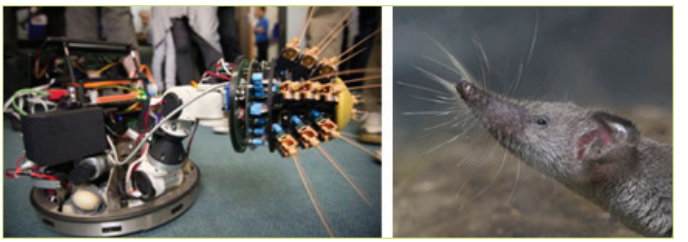
Disaster areas offer some of the trickiest types of terrain for robots or humans to safely navigate. Quadrupeds are one of the most adaptable platforms in situations like these which is one reason why we use trained dogs to search for earthquake survivors. Problem is, while dogs are great at getting around over the top of rubble and finding places where humans might be buried, they're too big to get down into nooks and crannies. Even if they could fit down in there, you probably wouldn't want to send them.

This kind of dangerous work is just exactly what robots are around for and by giving them rides on trained disaster dogs, they can get exactly where they need to go quickly and safely.

Carnegie Mellon University's Biorobotics Lab teamed up with Ryerson University's Network-Centric Applied Research Team (NCART) Lab and a (very well trained) dog named Freitag for a demo. While Freitag was lucky enough to be running around with a robot snake this time, the system can be adapted to deploy just about anything and apparently, deployment is controlled by the dog. Whenever it starts to bark (which it does when it smells a human), the robot jumps out of the dog's chest-pack and starts exploring.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

TAMING OF THE SHREWBOT



Back in 2008, the Bristol Robotics Lab in the UK had just finished building Scratchbot — a whiskery robot inspired by rats. Since then, they've been hard at work bewiskering more robots, including their latest creation: Shrewbot — inspired by the small (but capable) Etruscan shrew. The Etruscan Pygmy shrew is the smallest mammal on the planet at just a few centimeters long, and weighing under two grams. Despite their diminutive size, they're ravenous little powerhouses, requiring twice their body weight in food per day. To catch all the insects they need to eat to keep themselves functioning, shrews rely on their exquisitely sensitive whisker arrays to locate munchies in near total darkness. Jealous roboticists have attempted to duplicate the shrew's ability to adapt to environments in which visual systems don't work that well by creating robots that also use whiskers to navigate. Bristol Robotics Lab's Shrewbot seeks to mimic this method of "active touch" sensing and navigation through an artificial "shrew nozz" of sorts. Like a shrew, the robot can move its snout around independently of its body, and if any of the artificial whiskers brush up against an object or surface, the bot can instantly home in on that spot. The idea here is that a robot with whiskers can putter around environments that are dark or full of smoke, and not worry about getting lost or running into things. Of course, it's possible to do similar navigating without physical distance detectors, but whiskers have the advantages of being very cheap and very reliable, and no matter how fancy your sonar or LIDAR is, it's hard to do better than detecting an obstacle by interacting with it directly through touch. The other advantage of physical interaction is that sensitive whiskers can return much more information than simply where an object is. Apparently, shrews can sense things like shape and texture, and while Shrewbot can't quite get that much detail out of its artificial whiskers, they're working on it. The other part to all this whisker research is that it goes nose-in-hand with other forms of robotic touch, like fingers. While nobody's suggesting that we re-invent the finger (or the shrew, for that matter), there are certainly a lot of applications where having some whiskers would be preferable to having some fingers. For example, trying to touch something that may not particularly want to be touched. Whiskers are easy to replace. Fingers, not so much.

COMBAT ZONE

Discuss this section in the SERVO Magazine forums at http://forum.servomagazine.com

Featured This Month:

www.servomagazine.com/index.php?/magazine/article/march2012_CombatZone

BUILD REPORT:

Shaka v.3 – Part 1 Design Goals and Methodology

● by Thomas Kenney

Shaka Version 2 in Hindsight

One of the largest bots among the MH fleet and the deadliest of all the bunch, the first complete rebuild of our 30 lb Featherweight, Shaka (full build report in the October '09 *SERVO*) kept the basic layout of our first Featherweight while improving every aspect of the design. Before even considering investing time, money, and spare parts, what aspects justify a full redesign? And contrary to that, what actually did work?

Though the 7 lb weapon and accompanying parts took up a significant portion of the robot in both volume and weight, it also sported a pair of 18V OS DeWalt drive setups capable of shooting the bot across the arena at impressive speeds, and hardened 1/8" 4140 steel sloped 70° along the side for armor. Ablative pieces of 1/2" UHMW protected the front when fighting other spinners, alongside a set of steel skids that fed any wedges straight into the weapon. The light weight of the Lithium Polymer batteries helped fit all these in, providing all systems a reliable 22.2V, and in later competitions, a separate 37V to the similarly efficient Axi brushless motor — itself responsible for bringing the weapon up to such dangerous speeds.

That efficiency was countered by the frame layout and construction. While it did take up much more weight and space than it should have, this integral aspect of the design would eventually prove to be the weakest link; not just in the structure itself, but also the implications of complete non-invertability. Both were evident early in its robotic career, and the

second became more painfully obvious as I lost one particular high-stakes fight out of the blue from a single flip. Regardless, I held off this redesign for the time being. This was primarily due to the monetary costs involved, but at the same time most of the robot worked spectacularly. I opted to get the most out of my money, and continued to compete with Shaka — all the while second-guessing myself as the hits kept piling up.

Following almost two years, plenty of hits taken, and even more handed out, Shaka was finally retired following Motorama 2011. There, it went out in a blaze of glory ending with a full drive and weapon failure caused by the tweaked frame and baseplate shattering one of the DeWalt gearboxes — accompanied 30 seconds later by the other side's motor armature frying due to the weight strain. Then finally, the weapon motor rattled loose due to a poor mounting structure and hastily-applied thread fastener.

The aim with this subsequent redesign was to address these and all other issues, while at the same time keeping the things that worked which allowed Shaka to lead so many opponents into their own forced retirements.

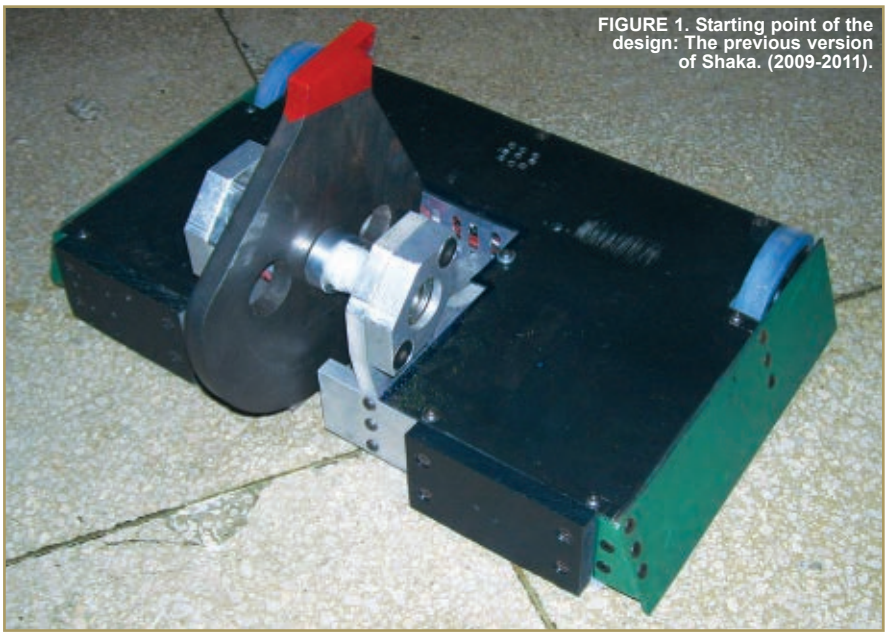


FIGURE 1. Starting point of the design: The previous version of Shaka. (2009-2011).

Primary Manufacturing and Frame Assembly

The targeted event for Shaka's redesign was RoboGames 2011 — less than two months after the previous version's final competition. In any other case, I would have opted out of the build completely due to the time strain. However, some certificates won at Motorama were enough for me to afford offloading some of the manufacturing, and have the 2D profiles of the frame and weapon waterjet cut by Team Whyachi.

After a few hours of work drawing up the parts in Solidworks (**Figure 1**), a week of waiting, some miscommunications on S7 thickness tolerance, and one small waterjetting error, I had all the basic parts for a new frame and weapon assembly cut from .5" 6061 aluminum and S7 steel of the same thickness.

The previous frame's construction involved mostly aluminum and UHMW rails sliding together in milled slots before being fastened, in an effort to relieve the screws and transfer any force to the nearby frame pieces. With this frame's parts being jet cut, it was more efficient to base the design around a tab-based construction with slots extending the full thickness.

In the final design, all vertical tabs were half the width of the frame rails, with at least one 1/4"-20 flathead screw above and below. Two triangular corner brackets slotted into the front and inner weapon rails, supporting the otherwise purely rectangular chassis structure and providing a solid slot/clamp mounting for the S7

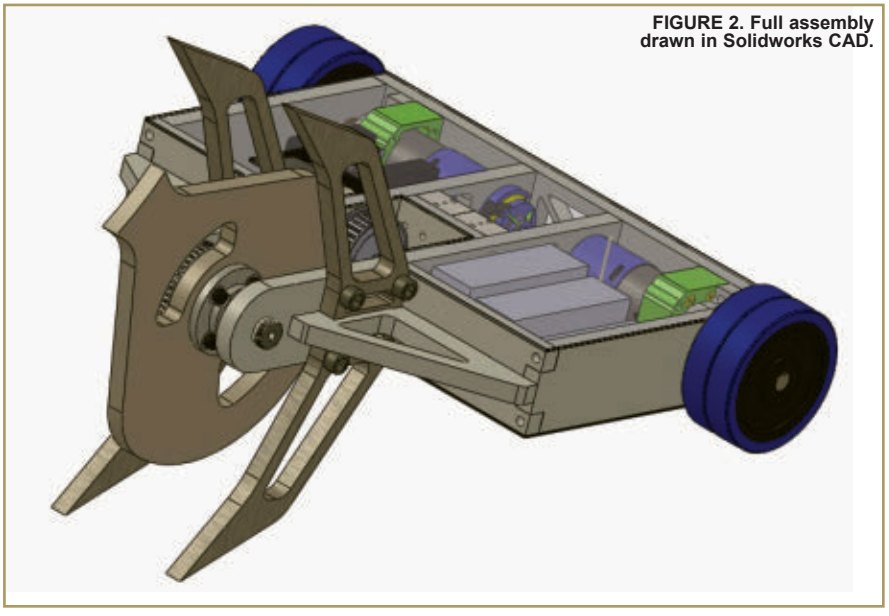


FIGURE 2. Full assembly drawn in Solidworks CAD.

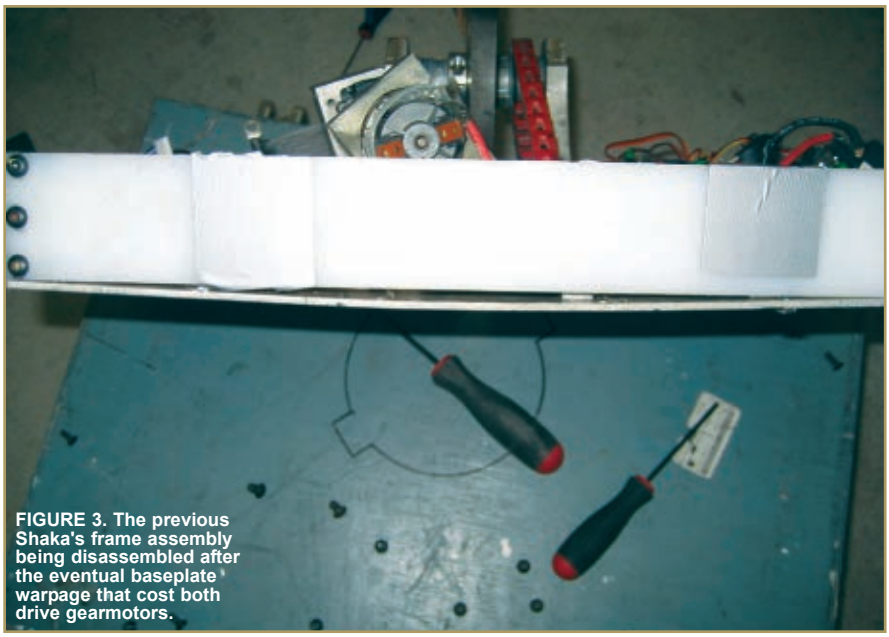


FIGURE 3. The previous Shaka's frame assembly being disassembled after the eventual baseplate warpage that cost both drive gearmotors.

front skids (which themselves remedied the previous non-invertability).

All of the slots were drawn as .01" oversize in each direction to account for material and waterjetting tolerances, though most still required some minor filing due to the .02" radius from the waterjet nozzle. After this — and a few hours drilling and tapping the 22 1/4"-20 holes, as well as the eight 3/8"-18 threads for the skids — the frame went together effortlessly.

As mentioned, the old .1" 6061 aluminum baseplate had held up initially, but the self-attrition of repeated downward force through a few years of fights was eventually enough to result in it and the frame warping to disfunction. Unlike that previous chassis — where the drive motor's mounting and some other critical structural points depended on the baseplate — this new .5" aluminum frame was entirely self-sufficient. This change, as well as the smaller footprint allowed for a significant weight savings and made the previous amount of armor not as necessary.

With no structural reliance needed, the new top and bottom were both cut from .1" polycarbonate. They have proven durable enough with all the proper mounting hardware in place and serve primarily as a gut-retainer rather than any structural purpose.

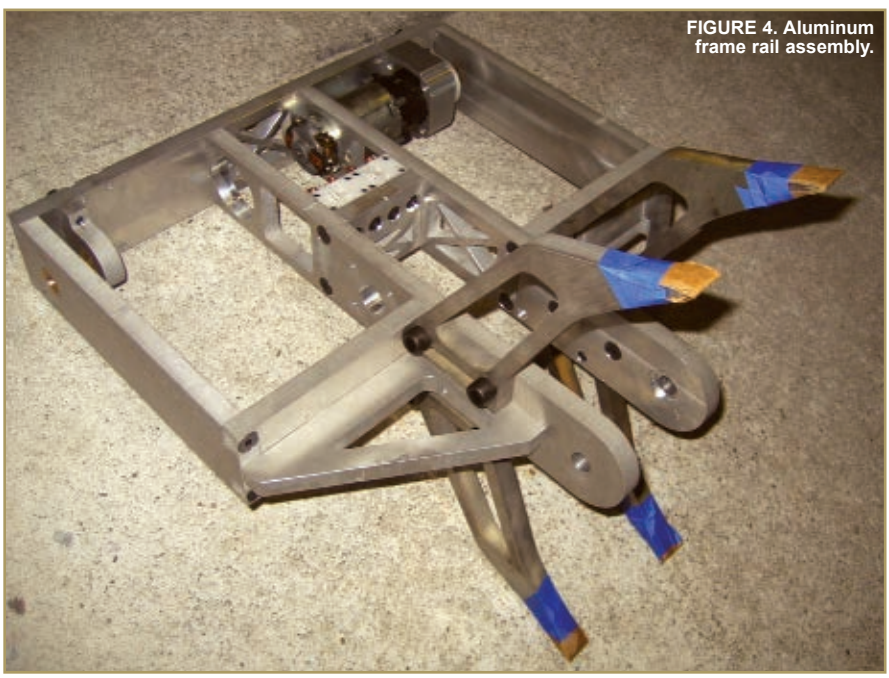


FIGURE 4. Aluminum frame rail assembly.

Integrating the Frame and Weapon Structures

Shaka v.2's baseplate dependency contributed to a lack of rigidness and structural integrity throughout. With the disc extending through a slot through the .1" baseplate and relatively flimsy .5" UHMW used for the three rear frame pieces, a single aluminum rail was the only continuously rigid structure running from side to side of the frame. In addition to this, the live .75" shaft itself often had its own issues, usually bending after a few good hits and resulting in an unreliable spinup, forcing it to be replaced every few events.

The alloy of the shaft was changed after the first of these incidents from an unhardened 1144 alloy, several unknown grades, and then finally some 4130 chromoly that I keyed using my mill. Though there were some issues with the hardening of the first 4130 batch, it eventually proved more adequate than the previous. It survived for several brutal fights at the robot's final event before eventually warping when I face-planted Shaka into the steel arena bumper during testing at its last competition.

The new design utilized case hardened shafting from McMaster-Carr — the same .75" diameter as the previous design — now in a dead shaft configuration for more rigidness in the front portion of the frame. The simplest way to achieve

this setup would be to screw in the shaft from each weapon rail. Rather than risk relying solely on the shear strength of the 3/8" screws, I opted to build on this basic setup with a more complex assembly.

Each side used a 3/8" screw with a 1/2" diameter, 1/2" length shoulder that sat snug in the unthreaded 1/2" thick aluminum. On the inner side of the rails, a 3/4" diameter hole was bored 1/8" deep concentric to the screw's positions. The shaft sat in this inset, and another 1/2" bore was cut into each side of the shaft to contain the additional 1/8" of shoulder. All this and the shortened shaft length (2.5" from 4.5" between the rails) created a rigid assembly that spreads any impact force equally between the frame and the shoulder portion of the screws.

To work alongside the dead shaft, I needed to redesign the weapon disc assembly with an in-bearing. I happened to have the weapon hub/pulley setup last used on the 2009-2010 version of Sloth — another vertical disc wielding 30 lber the previous version of Shaka had forced into retirement. The setup consisted of a CNC'd disc on one end, opposing the thick 5 mm pitch aluminum pulley, both pressed over a lengthy .75" inner diameter oillite bushing. The assembly clamps around the weapon disc, tightened by a quartet of 1/4"-20 screws. The long bushing covered all but .1" of the case hardened shaft and provided stability for the massive flywheel on all axes.

When drawing up the new disc, I designed around this mounting pattern and was able to fit the weapon on the used hub with much less complication than the previous live shaft and keyways. It shared the same 11" diameter, .5" thickness, and other basic dimensions with the previous Shaka's primary weapon, including the counter-balanced single-tooth which was now sharpened in both directions for complete invertability should an adequate reversible BLDC speed controller become available. Both of the old discs would later be bored and drilled to serve as spares.

Powering the 40 tooth weapon end, the brushless motor's 32 tooth pulley provided for a 1.25:1 reduction, limiting the weapon's speed to around 7,500 RPM at 37 volts. Unlike most of my designs, I already had several of the 5 mm pitch belts at the time of the CAD work, and was able to design the weapon and motor mounting distance around the existing setup. Due to some slight rounding of the digits, this still turned out off, and some sort of tensioning unit was needed to keep the belt running tight enough to be efficient (**Figure 6**).

FIGURE 5. Shaka v.3's weapon disc (larger of the three) assembled with the hub/pulley hybrid.

Fortunately, the S7 skid's 3/8" mounting screws provided an optimal point to attach a tensioner. This last-minute addition consisted of some 1" UHMW stock bored for a 3/8" oillite bushing. We replaced one of the four screws with a 1.75" long model, moving the head to the inside of the frame and adding a locknut on the skid end; the added length of the screw was

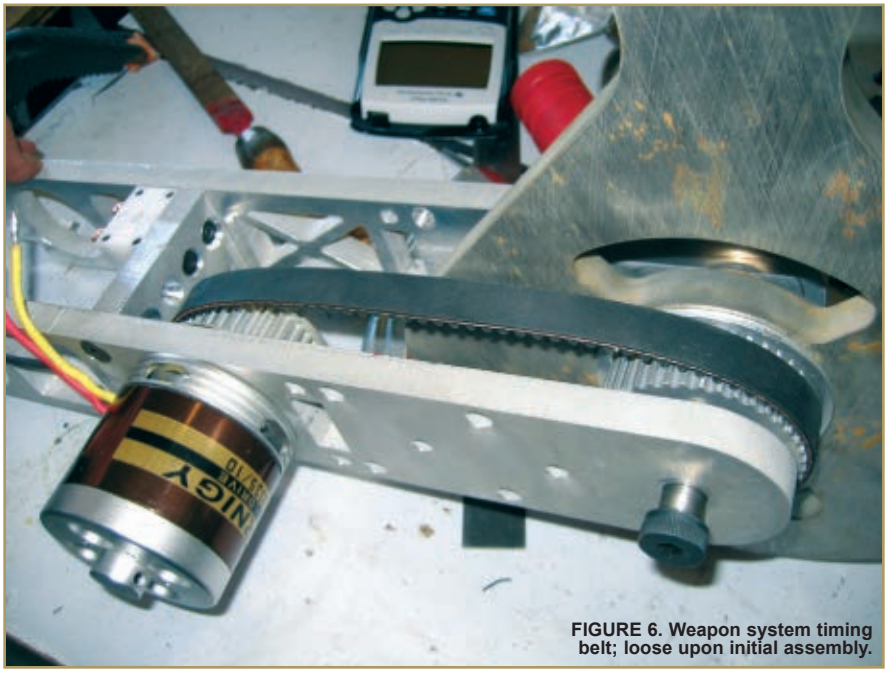


FIGURE 6. Weapon system timing belt; loose upon initial assembly.

FIGURE 7. Improvised tensioning roller applied to the belt.

unthreaded, allowing the bushing to spin smoothly. Lastly The tensioner was counterworked to recess the screwhead (**Figure 7**), making for a compact, efficient assembly that keeps the belt running true.

This build of Shaka uses the same Axi 5330/18 brushless outrunner motor to drive its weapon disc. Because of the build of the motor, when mounted only by the threaded holes in its face the spinning can and — in effect — much of its weight goes unsupported. Enough impacts can lead to the shaft bending, expensive magnets shattering, and other incapacitating damage.

The idea of a support bracket was always on my mind when I designed this version's frame, but with weight concerns, and the rushed design process I had to opt out of including it. In the final stretch of the build marathon after I had already pocketed out the side aluminum for weight, we found two ounces to spare and decided to make use of them to protect the $270 brushless.

On the end of the Axi's stock can, there is a thin nub (typically used for a prop attachment in RC planes) just under 1" diameter. A support block was cut from UHMW stock, then bored out and pressed with a 1" oillite which screwed to the front of the frame and sat tight between the top and bottom polycarbonate, providing a solid support for the rear of the outrunner.

Minor Electrical Changes

The majority of the robot's electrical guts were carried over as they had been in the preceding version. Drive and weapon are independent. A string of Turnigy 2,200 mAh 40c LiPo packs totaling 10s1p (37V) powered the weapon, while a single 6s1p (22.2V) pack of the same brand and size was used for the two drive motors. Each setup was wired to be powered on and off by a dedicated Whyachi MS-05 switch.

At the time of the design, the intended motor controller for the weapon was a Jeti-90a. For our purposes, any of the wide range of similar sized brushless controllers would suffice, though I am still hoping for the eventual development of a reversible model with the ability to handle the system's full 37V, and — as mentioned — design the weapon with that in mind.

Choosing the drive ESCs is still a bit more difficult with the lack of workable options in the range of 60A continuous reversible brushed controllers. The previous builds of Shaka had used a set of the standard Victor 883s to control the 18V DeWalt drive system. Over the last four years, I've had almost a dozen Victors go up in smoke — some from their lack of current limiting and others with no plausible explanation when the motor's continuous current draw was nowhere near the peak amperage.

With the lack of options in this robot size range, I had planned on wiring up another pair but as the scale readout hovered at 30 lbs at the end stretch of the build, I eventually decided to try out a single Holmes Hobbies BR-XL brushed controller I had acquired. The BR-XL is rated for 80A continuous compared to the 883's 60A (with fan), and outputs a PWM signal of normal strength — all at less than half the weight.

Being a last minute addition, the BR-XL was powered in parallel with one of the remaining 883's due to availability, though I've since run a full pair of them in a comparable drive setup with no issues. A detailed review on the ESC by Mike Jeffries is available in the May '11 *SERVO*.

Although the ability to afford outsourcing some of the initial fabrication, a better organized design layout, and a few more years of general experience all improved the efficiency of the build, three weeks was still a tight schedule.

Now that I've covered the machine's technicalities and the reasons behind each, I'll continue next time with the build process itself, and what it took in the attempt to bring the third version of Shaka from concept to creation in that short a time. **SV**

BUILD REPORT:

Antweight – Motor City Massacre

● by Mike Jeffries

Motor City Massacre was a departure from my normal design method. Most of my previous bots have been constructed to have the biggest weapon or thickest armor I could fit within the weight limit. With Motor City Massacre, instead it was built around the weapon concept with the knowledge that once it was all together and working, a substantial amount of weight would need to be removed.

Motor City Massacre was designed to be made entirely on a waterjet with commercially available hardware used to connect the various panels together. The main structure uses a mixture of hex standoffs and nutstrip (available at **kitbots.com**) to hold the flat panels in place. Assembly went very fast, however, it was quickly becoming apparent that the weight issues were going to be severe.

After the initial build was completed, it was obvious that drastic measures were needed to get Motor City Massacre under weight. The 1/32" steel armor that was supposed to be used on both sides was removed from the side covering the servo. This switch meant that none of the electrical system could be placed in that half of the robot.

Buried beneath the tangle of wires are two Fingertech TinyESCs, two 22.2:1 Silver Spark gearmotors, a HobbyKing R410 receiver, a battery elimination circuit for servo power, and a custom hex wrench operated power switch.

Motor City Massacre has already attended its first event. It was driven to a 3rd place finish at Dragon*Con Microbattles without the use of the grabbing claw since the mount had not yet been finished. During testing after the event, I noticed that the robot was bouncing at high speeds. I found that the Lite Flite wheels were not particularly round and were the source of the bounce. To correct this issue, I put a 3 mm

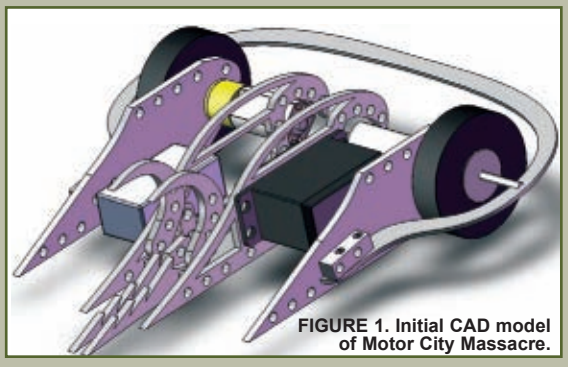


FIGURE 1. Initial CAD model of Motor City Massacre.

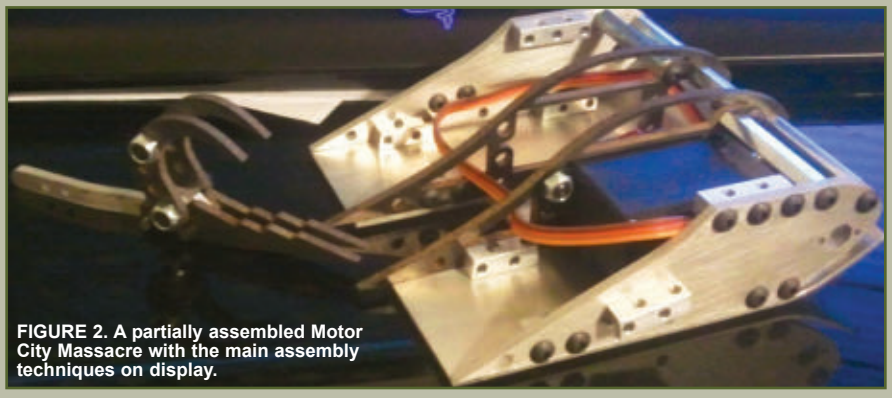


FIGURE 2. A partially assembled Motor City Massacre with the main assembly techniques on display.

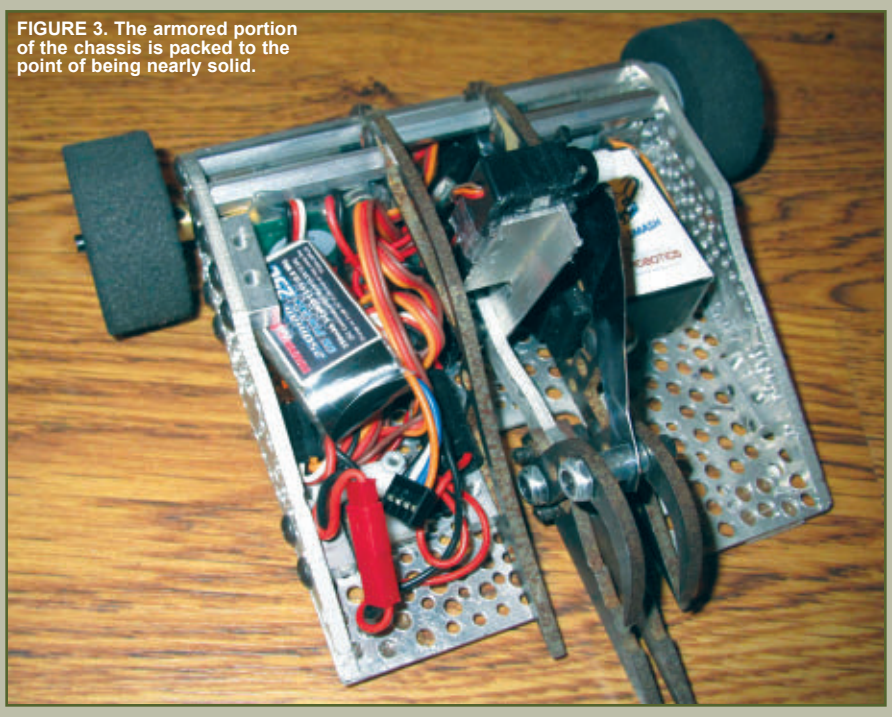


FIGURE 3. The armored portion of the chassis is packed to the point of being nearly solid.

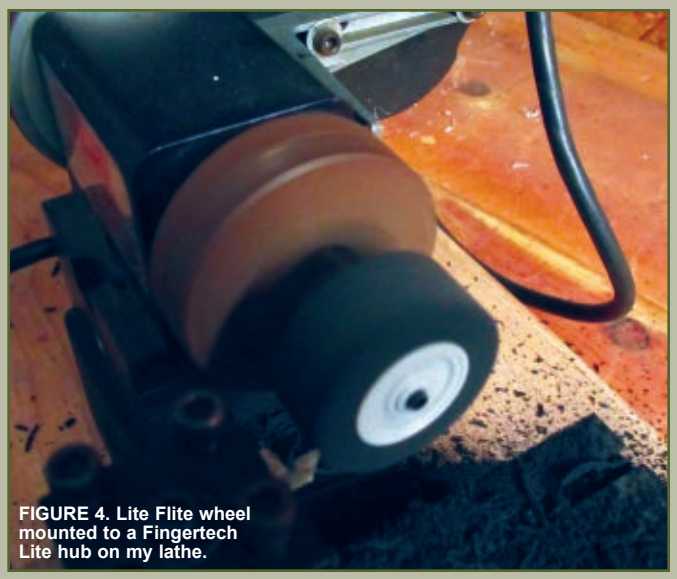


FIGURE 4. Lite Flite wheel mounted to a Fingertech Lite hub on my lathe.

FIGURE 5. Motor City Massacre fully assembled and lightened with generous application of a drill and rotary tool.

shaft in the chuck of my lathe and ran a tool across the surface until the surface no longer varied relative to the shaft. Doing this has greatly improved how the robot drives, and with less bouncing it seems to have better traction.

Even with the issues during the build and the drastic weight reduction measures, Motor City Massacre, has proven to be a fun and durable robot. Video of Motor City Massacre, as well as many of my other robots can be found at **youtube.com/mikencr**. SV

BUILD REPORT:

*Nyx, Sportsman, and Dragon*Con 30 lb Bot – Part 1*

● by Mike Jeffries

Motorama last year was surprising. I brought three robots and at the end of the event, only one needed to be rebuilt.

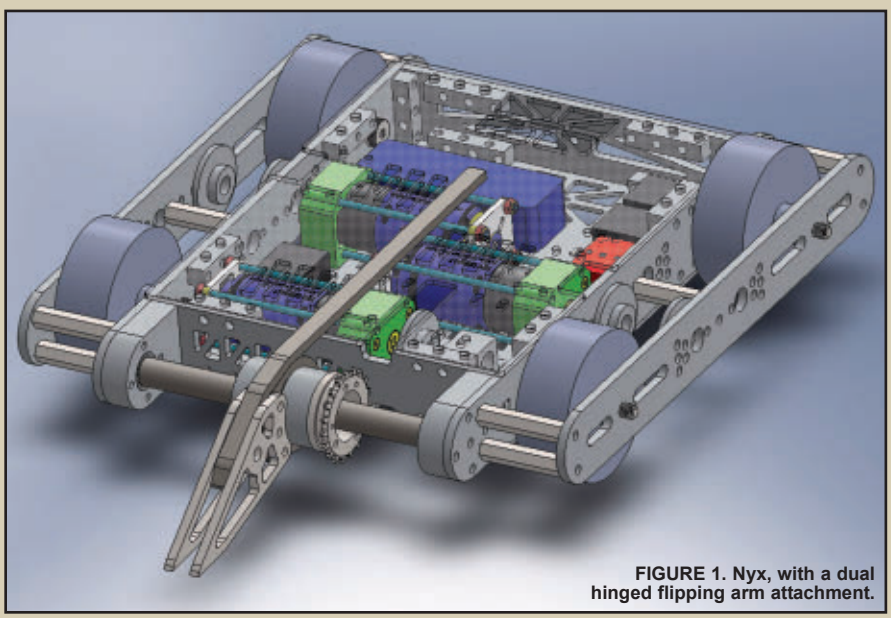


FIGURE 1. Nyx, with a dual hinged flipping arm attachment.

During the summer, I built a second Antweight for my teammate to drive, and we took all but Moros, my 30 lb bar spinner, to Dragon*Con Robot Battles. All of the robots survived Robot Battles with no major damage which meant no robot work to be done besides basic maintenance between September and February. Having that much free time before the next event led to the logical conclusion: I should build another robot. The question then was, "What do I build?"

There were a few factors that I had to consider. Which events would I take it to? What are the unique rules for each event? What do I have lying around that I can reuse to save a bit of cash?

Once all of these questions were answered, I decided on a

30 lb robot with an electrically actuated weapon, with interchangeable attachments.

The decision to do interchangeable weapons was based entirely on the rules of the events it would attend. At NERC (**www.nerc.us**) events, the Sportsman class requires active weapons and bans wedges and high energy spinning weapons which are limited to 400 RPM without specific approval. At Robot Battles, wedges are not specifically banned; however, the arena floor consists of stage risers with random obstacles added to interfere with low robots. They also ban spinning weapons with a tip speed in excess of 20 ft/s because there is no arena wall or barrier between the robots and crowd.

Based on these rules, the dual hinged flipping mechanism is intended to be the primary weapon option. However, the attachment design allows the entire weapon to be swapped by removing three bolts. The other options include a ramming spike and a rotary grinding/lifting wheel that — based on the motor RPM and gear reduction — should spin at 399.7 RPM; just under the 400 RPM limit, even without friction losses.

Component selection for this design was fairly simple, since I have a few components that have already proven themselves in combat to base many of the decisions on. The drive system will use two 18V Dewalt power drive kits in high gear that chain drive 4" wheels. The weapon is powered by another Dewalt in low gear. For motor controllers, I am using the Holmes Hobbies BR-XL controller on all three Dewalts with the weapon controller set to use an available active braking feature.

My power switch is a scavenged Whyachi MS-01 from my old 60 lb robot. The only experimental component is the battery. My normal battery supplier has suspended operations and with only three good A123 cells on hand, I decided to try something new. I'll be using Turnigy Nano-Tech 2,650 mAh batteries in a six-cell pack for power.

The majority of chassis components have just arrived and are about to be prepared for assembly. The vast majority of the work on these parts was done via waterjet by Westar Mfg. (**http://teamwhyachi.com/botshop.htm**) with only a few drilled and tapped holes required to piece the chassis together. The chassis was able to be made this way because I am using 1/2" nutstrip from Kitbots (**http://kitbots.com**) to hold most of the panels together. SV

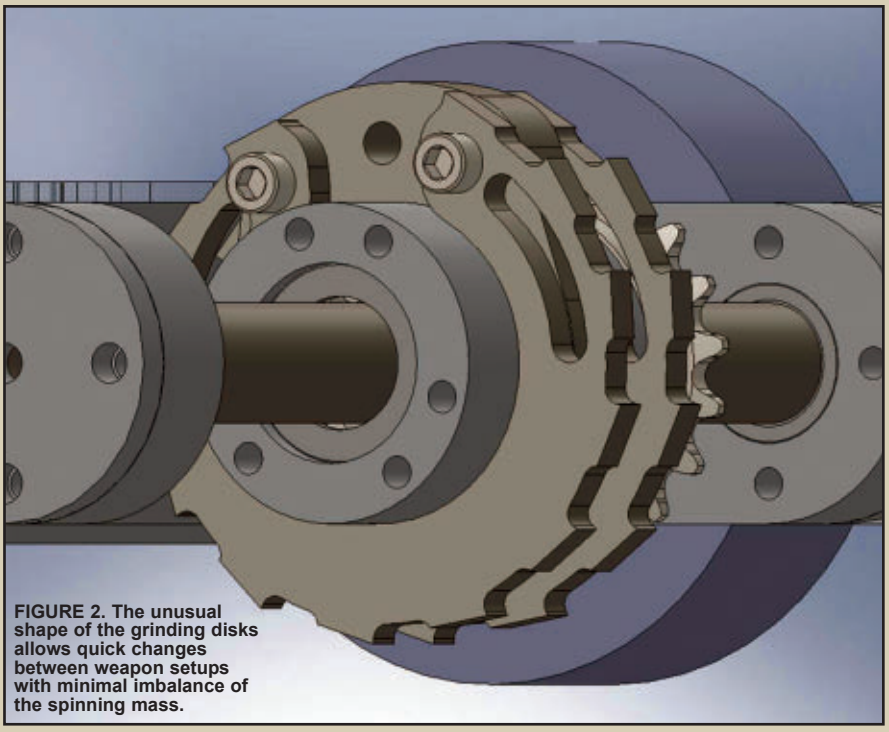


FIGURE 2. The unusual shape of the grinding disks allows quick changes between weapon setups with minimal imbalance of the spinning mass.

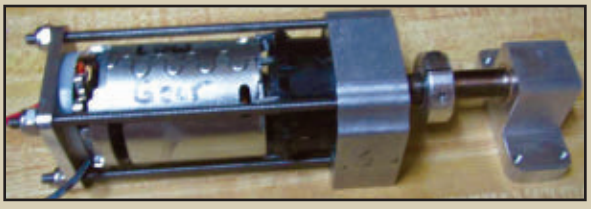


FIGURE 3. This is the weapon motor with a custom pillowblock made from scrap material.

FIGURE 4. The chassis fresh from machining and ready to begin the assembly process.

PARTS IS PARTS:

Making Wheels Round

● by Mike Jeffries

There are a decent number of ready to use wheels available today for robots of all sizes, and one of the most popular for small robots is the Lite Flite foam wheel. They're cheap, tough, and come in a wide range of sizes. They are made up of a two piece snap-together plastic hub and a foam disk. The low cost does have a few down sides, as I've found major consistency issues batch to batch with these wheels.

The main issue that will have an effect on a fighting robot is how concentric the tread is to the shaft of the axle it rides on. I have yet to find a Lite Flite that does not have at least some side to side wobble right out of the package. This may seem like a minor issue, but when you're spinning the wheel at a decent speed this wobble turns into bouncing which then turns into reduced control.

Another common issue is the tread surface being curved or conical which results in a reduced contact patch and less traction. To fix these issues, I've started turning my wheels on a lathe to provide a controlled and flat surface with a consistent diameter across several batches of wheels.

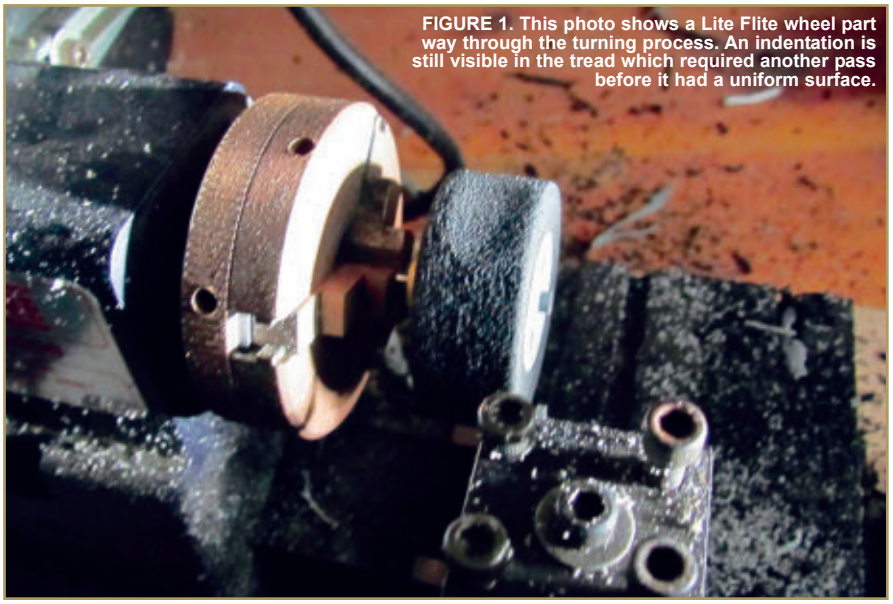


FIGURE 1. This photo shows a Lite Flite wheel part way through the turning process. An indentation is still visible in the tread which required another pass before it had a uniform surface.

If you have access to a lathe, the process is simple. Begin by attaching a cutoff of round stock that is the same diameter as the shaft the wheel will mount on in the jaws of the lathe. Once the axle is secure, slide the wheel on (most applications will require an adapter like the Fingertech Robotics Lite Hub to mount the wheel to a shaft) and secure it to the axle. At this point, you are ready to cut the wheel down to size.

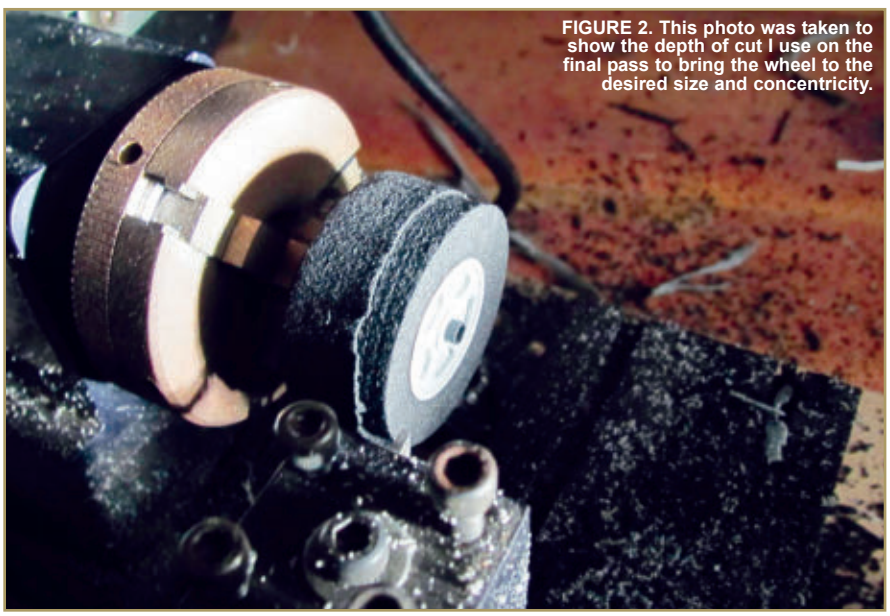


FIGURE 2. This photo was taken to show the depth of cut I use on the final pass to bring the wheel to the desired size and concentricity.

If you want to drastically alter the wheel diameter, fairly deep cuts can be done. However, the finishing cuts should be done with a shallow cut depth and at low travel speeds to provide the best finish. Shining a bright light at a shallow angle across the wheel will highlight imperfections and show you areas that may need a second pass at the same cut depth.

Cutting the wheel down this way leaves a rough edge on both sides of the wheel. An easy way to fix this is to carefully apply a file

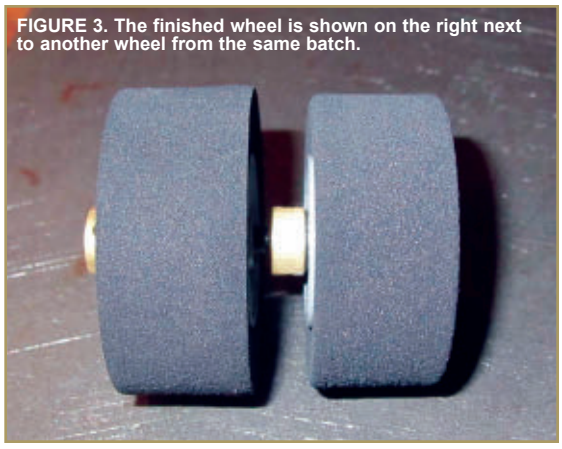


FIGURE 3. The finished wheel is shown on the right next to another wheel from the same batch.

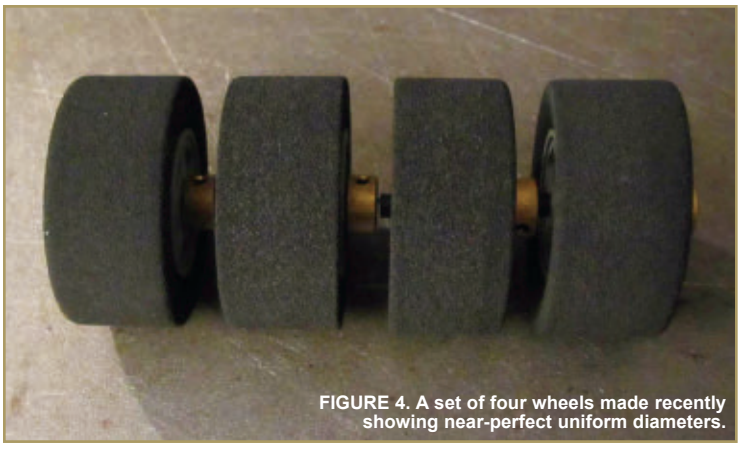


FIGURE 4. A set of four wheels made recently showing near-perfect uniform diameters.

while the wheel is still spinning on the lathe to give it a smooth, rounded edge.

If you don't have access to a lathe and want results like this, you'll need to get creative. Similar results could be achieved with a drill press, a steady hand (or a vise), and a sharp file. It may also be possible to build or buy something that functions like an RC car tire truer which uses a sanding wheel on a parallel shaft to sand down the tread to a decent finish at whatever diameter you desire.

With the proper setup, you can easily create a large supply of uniform wheels to use on your small scale robots in a short amount of time.

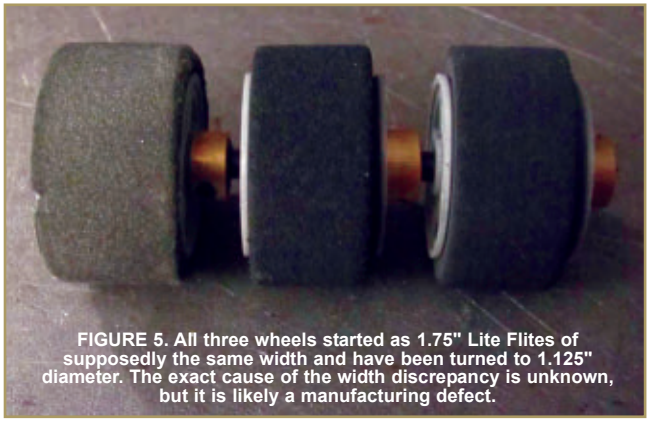


FIGURE 5. All three wheels started as 1.75" Lite Flites of supposedly the same width and have been turned to 1.125" diameter. The exact cause of the width discrepancy is unknown, but it is likely a manufacturing defect.

The main point of doing this to your wheels is to attain the best drivability possible with a particular drive system arrangement. Cutting the wheels down to a concentric surface and cleaning the edges results in wheels that have little to no bounce and the largest contact patches possible for their width.

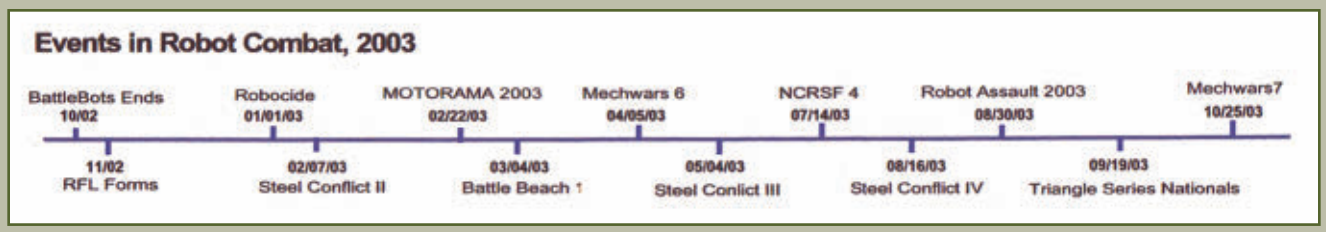
In robot combat, the little things can make all the difference and many matches have been lost by $4 parts failing to perform. Making these changes has resulted in a noticeable improvement in traction and drivability in both of my 1 lb robots. It may seem excessive to put this much effort into such low cost parts, but like I said attention to detail can make all the difference in the arena. SV

The History of Robot Combat: Life After BattleBots

● by Morgan Berry

After the cancellation of Comedy Central's BattleBots in late 2002, many in the building world felt lost. BattleBots had grown the sport of robot combat from an obscure niche activity to a national television event. Robot combat and BattleBots had become household terms. Suddenly, the backbone of the sport was gone, and that left many with questions as to the future of the sport. Would robot combat continue to survive without the support of those national television audiences?

The builders did not wait long to mourn the ending of BattleBots. In fall 2002, Ken Gentry began planning a competition in Orlando, FL. Many hoped that this event — Robocide — would be the next "big thing" in the wake of the ending of the BattleBots television program. This first major event since



BattleBots — quite appropriately held on New Year's Day in 2003 — attracted many well-known bots including Son of Wyachi and Phrizbee-Ultimate. To gain a perspective on this event, I sat down with Combat Zone editor at *SERVO*, Kevin Berry, to talk about his experience. Because this was his first ever competition, he was able to gain the perspective of a new "outsider" looking in on the veterans of the sport.

Kevin built a 55 pound bot, Chupacabra, which he describes as both "lame" and "pathetic," and unfortunately, was scheduled for the very first fight in the competition against veteran Dick Stuplich's 2EZ. His first experience in the box lasted approximately 20 seconds, ending with 2EZ turning the poor Chupacabra into — in Kevin's words — "coleslaw."

This harsh induction into the world of robot combat taught Kevin something that surprised him: Watching your own bot get destroyed is just — or at least almost — as fun as destroying someone else's. Really, although winning is of course ideal, robot combat is just all around fun to be a part of. He finally understood those moments he had witnessed on BattleBots where a driver cheered on his competitor as his own robot was torn apart. Of course, that spirit would be nowhere to be found without the good sportsmanship of the competitors which Kevin says was the other lesson he learned that day.

As he met the more famous faces of robot combat, every single person was "gracious, informative, and approachable." Everyone at Robocide did their best to welcome Kevin into the sport which he would, in turn, eventually do for the other "newbies" he met. The things he experienced at Robocide were very indicative of the spirit and manners of robot combat.

All in all, the event was a great success and set the tone for 2003 to become a very active and positive year in robot combat. If you would like to see videos from this event, Robert Woodhead has a great catalog of the fights at Robocide on his website at **www.madoverlord.com/robots/robocide.t**.

Robocide was among many regional events held in 2003. As Terry Ewert from Team Wyachi puts it, "It seemed like we were on the road every month, from Harrisburg and Pittsburg, PA, to Orlando and Ormond Beach, FL, to Minnesota and Saskatoon in Canada, to Los Angeles." Along with this myriad of smaller events, there was also an effort to create a national organization.

Battle Beach was held in Daytona Beach, FL on March 4th, 2003. Brian Nave, the event host, intended the event to correspond to Bike Week — a 10 day festival for motorcycle enthusiasts. This competition featured a unique arena design. Since it was made to fit on a curved stage, the arena was hexagonal, long, and narrow, rather than being fairly square like most other arenas. This made for some interesting fights, as the speedier bots had plenty of room to get up to full speed in the long arena. Terry Ewert went so far as to say the best match he has ever seen took place at Battle Beach. This fight — Toro versus The Judge — went the full three minutes with no knock-outs and was almost non-stop action. Both 340 lb bots were flung into the air at times during the fight.

The competition was very well attended; 106 total fights were held. Nora Judd, a competitor at Battle Beach, repeated Kevin Berry's message about the camaraderie at all the events in this time period. Brian Nave was short on helpers (she explained to me), so many people stepped in to help run the event. She ran the pit area and her husband, Steve Judd, was in charge of organizing judging. Another Steve — Steve Buescher — set up the schedules while yet another — Steve Brown — helped out, as well. Battle Beach was held annually through 2006, with subsequent battles featuring large insect battles (for more information, stay tuned for next month's article on "The Rise of the Insects").

Battle Beach served as the Southeastern qualifier for a larger, national competition. The national competition — known as the

Triangle Series Nationals — was sponsored by the Robot Fighting League. Finalists from Battle Beach, Mechwars (the Northern qualifier) and Steel Conflict (the Western qualifier) competed in Minneapolis, MN on September 19, 2003. Nora Judd (who was the first female driver to make it to Nationals) again recalled how the tireless efforts of numerous builders came together to create the large event. Although she admits it was disorganized — as so many robot combat events are — everyone pitched in to make the Triangle Series Nationals a success.

In addition to the events held during 2003, there was also growth on the administrative side of the sport. The Robot Fighting League was formed in November 2002 by a group of builders who wanted to standardize and expand robot combat. Steve Judd, Fuzzy Mauldin, Bob Pitzer, and Steve Brown were among those who attended a highly secret meeting at an event in Las Vegas. It was so secretive, in fact, that even Nora Judd (wife of RFL founding member Steve Judd and an active robot combat participant) was kept in the dark about it. They met out of necessity; as Nora explains, "BattleBots had been cancelled and everyone was having so much fun. We wanted to keep playing! We didn't know exactly where we were going, but we knew that we wanted to continue."

What formed out of that secretive first meeting was a loose confederation between independent events held all over the world. RFL sanctioned events were subject to a set of rules and safety standards. These new standards arose to fix the issues that some builders saw with the BattleBots regulations (which couldn't legally be used outside of a BattleBots sanctioned event, anyway). Because of the high number of RFL sanctioned events (there were over 2,000 matches at RFL events in 2003 alone), builders could be assured consistency in the competitions they attended. The RFL also employed a system where new event organizers were sponsored by a more experienced organizer to ensure safe and successful matches.

In the year immediately following the ending of the BattleBots television show, there was a definite change in robot combat; 2003 was characterized by a return to the grassroots, homegrown spirit that originally sparked the creation of robot combat. An inclusive, tight-knit group of builders were working to ensure that the sport was not going to fade away. In fact, in many ways, it was better than ever.

For more information, check out the timeline of some of the major events in robot combat in 2003, or go to **Botrank.com** or **Builders db.com**. Special thanks to Kevin Berry, Terry Ewert, and Nora Judd for the wealth of information they were able to provide about this period in robot combat.

Next month in our series will be The Rise of the Insect: Antweight and Beetleweight Competitions. SV

***Note:** Unfortunately, the builders of this time period seem to have done a spotty job at recording the events of the day. As a result, I had a difficult time piecing this article together. I did my best to dig up all the information I could, but if any readers notice anything I've left out or misconstrued, please contact SERVO so I can set the record straight.*

EVENTS

Upcoming and Completed Events

Upcoming Events for Mar-Apr 2012

The Central Illinois Bot Brawl 2012 will be presented by the Central Illinois Robotics Club in Peoria, IL on March 24th. For further information, go to **http://circ.mtco.com**.

The 2nd Annual Downtown Dogfight will be presented by PennBots at Harrisburg University in Harrisburg, PA on Saturday, March 24th. For further information, go to **www.pennbots.org**.

RoboGames 9 will be held April 20-22 in San Mateo, CA. For further information, go to **www.robogames.net**.

STEM TECH Olympiad 2012 will be presented by the United States Alliance for Technological Literacy in Miami Beach, FL on April

25-29. For further information, go to **www.usatl.org**.

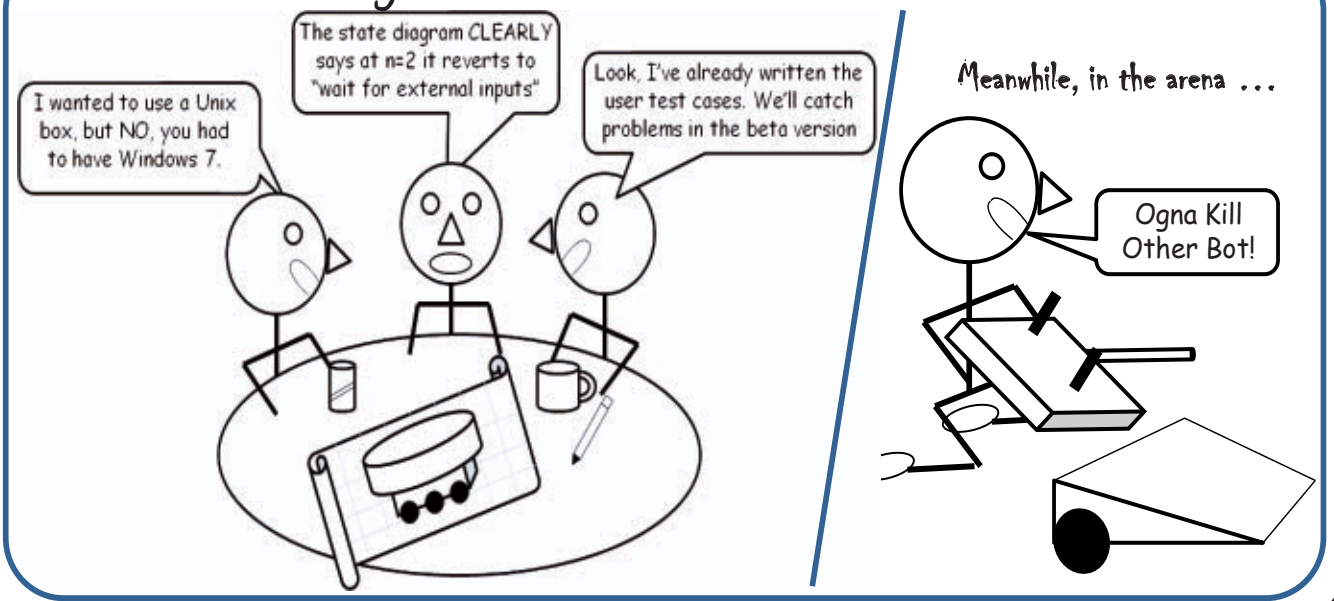
Completed Events for January 2012

Gulf Coast Robot Sports 10 was presented by Gulf Coast Robot Sports in Bradenton, FL on January 14th. SV

Melty Brains

by Kevin Berry

Why Combat Bots Aren't Autonomous



Sounding Off

Adding Sound to Your Arduino Robot

by Gordon McComb

Discuss this article in the *SERVO Magazine* forums at **http://forum.servomagazine.com**.

My first robot with sound used the mechanism from an old cassette recorder. Music and sound effects were stored on short tapes. The whole thing was mechanically driven with a solenoid that turned the PLAY button on and off. The biggest problem: There was no way to know what musical jingle the robot was going to play next.

Thanks to several ready-made add-on boards for the Arduino and other popular microcontrollers, you can incorporate melodies, effects, and other sounds to your robots. You have full control over what's played — from short quarter-second gunshots to 10 minute lullabies.

In this article, you'll read about how to give your mechanical creations completely electronic sound effects, music, and even voice. The subject of robotic sound is a large one, and a single installment isn't enough to address everything. So this time around, I'll cover basic sound generation using just the Arduino — with and without external amplification — plus adding low cost effects, music, and voice co-processor chips to simplify the noise making.

Next time, I'll cover even fancier sounding off methods, including playing WAV, MP3, and other digital files, making music with MIDI, and building your own sound co-processor.

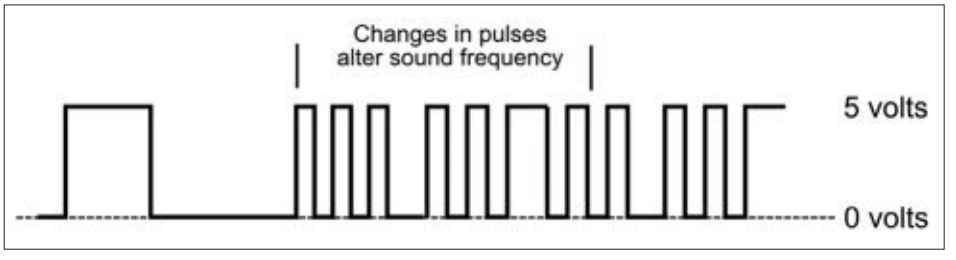
Important note: The demo code in this article is intended for the Arduino 1.0 integrated development environment (IDE). If you haven't already, be sure to download the latest Arduino IDE software. You can install it side-by-side with any previous version you already have. This is recommended, since some Arduino examples you'll find on the Web may not yet be compatible with the 1.0 software.

The Sound of Arduino Music

Any microcontroller with a pulse width modulation (PWM) feature can be used to produce sound effects and music. PWM is a series of pulses as shown in **Figure 1**; when the pulse frequency is within the range of hearing — about 20 Hz to 20 kHz — it comes out as a tone. The sound is heard by passing the pulses by way of the microcontroller's I/O pins through a speaker or amplifier.

By varying the frequency of the PWM, you make high, middle, and low tones. You can produce music with one PWM output (that's called *monophonic*), or you can combine the outputs of two or more PWM

FIGURE 1. Pulse width modulation is a common technique for making sound using a digital output. By using pulses within the range of human hearing (about 20 Hz to 20 kHz) and changing the timing of the pulses, the signal is heard as a sound tone.



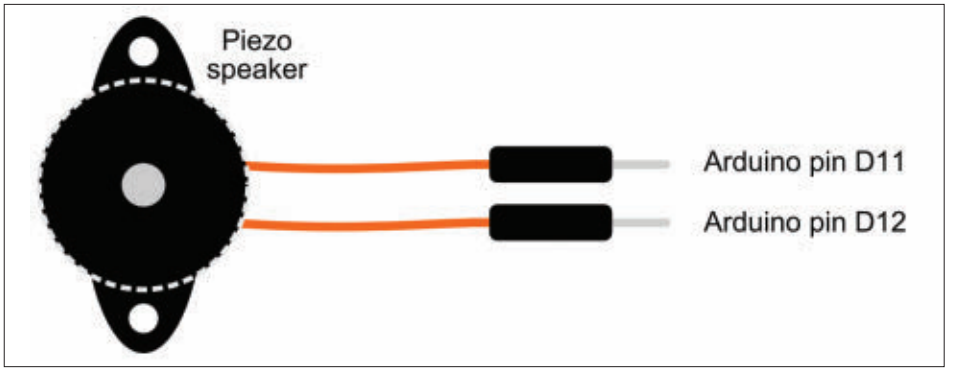


FIGURE 2. Attach the piezo element directly to the Arduino I/O pins. To simplify the connection (so the two wires from the speaker are next to each other), I'm using one of the pins as ground. Refer to the sketches for how this is done.

I/O lines to create *polyphonic* sounds, like a music synthesizer. With PWM, it's easy to produce warning sirens, warblers, bio-sounds (a la *R2-D2*), and other effects.

Sound With a Piezo Speaker Element

Figure 2 shows the basic connection diagram for a piezo transducer as a sounding element, connected to the Arduino. Piezo speakers are preferred (for getting started, anyway) because they have a high input impedance, and so they won't draw a lot of current from the Arduino's I/O connections.

The Arduino IDE comes with several tone-making examples you can try. To get things started, I've boiled down the code to the core concepts to make it easier to follow and adapt. Check **Listing 1** for a simple siren demo. It works by alternating two frequencies: 440 Hertz (Hz, or cycles per second) and 880 Hz; once every half second.

In case you're interested in such things, 440 Hz is concert A pitch (the A above middle C on a piano); 880 Hz — also an A tone — is exactly one octave higher.

An enhanced tone-making sketch is provided in **Listing 2**. It's adapted from the Tone example at **arduino.cc/en/Reference/Tone**, but is stripped down to make it simpler to follow. The sketch cycles through two sets of short musical notes. The *frequency* of each note is specified as one of four defined constants; the *duration* is a numeric value representing a fraction of a second: 1/1 is one second, 1/2 is a half second (and therefore a half note), and so on.

Figure 3 shows how the piezo speaker is partially tucked under the Arduino board on the ArdBot expandable robot; this expandable and affordable platform is detailed in the November '10 through May '11 issues of *SERVO*.

LISTING 1 – ArduinoSiren.ino

```
void setup() {
  pinMode(11, OUTPUT);       // Speaker connected to pins 11, 12
  pinMode(12, OUTPUT);       // Use pin 12 as speaker ground
  digitalWrite(12, LOW);
}

void loop() {
  tone(11, 440);             // Concert pitch 'A'
  delay(500);                // Wait 1/2 second
  tone(11, 880);             // Octave higher
  delay(500);                // Wait 1/2 second
}
```

LISTING 2 – MakeTones.ino

```
#define SPKR       11
#define SPKR_GND   12
#define LED        13

#define NOTE_C3   131
#define NOTE_B3   247
#define NOTE_C4   262
#define NOTE_D6   1175

void setup() {
  // Piezo speaker connected to pins 11 and 12
  pinMode(SPKR_GND, OUTPUT);       // Ground for speaker
  digitalWrite(SPKR_GND, LOW);
 pinMode(LED, OUTPUT);
}

void loop() {
  beep_beep();
  delay(2000);
  beep_beep_beep();
  delay(2000);
}

void beep_beep() {
  int tones[] = {NOTE_D6, NOTE_C3};
  int toneDurations[] = {2,4};     // Half, quarter note
  makeTone(tones, toneDurations, sizeof(tones)/sizeof(int));
}

void beep_beep_beep() {
  int tones[] = {NOTE_C4, NOTE_B3, NOTE_C4};
  int toneDurations[] = {8,4,1};   // Eighth, quarter, full note
  makeTone(tones, toneDurations, sizeof(tones)/sizeof(int));
}

void makeTone(int tones[], int toneDurations[], int length) {
  digitalWrite(LED, HIGH);       // Turn on LED when making tone
  // Iterate notes of tune
  for (int thisNote = 0; thisNote < length; thisNote++) {

    //Calculate the note duration
    int toneDuration = 1000/toneDurations[thisNote];
    tone(SPKR, tones[thisNote],toneDuration);

    //Add slight pause between notes
    int pauseBetweenNotes = toneDuration * 1.30;
    delay(toneDuration * 1.30);
    noTone(SPKR);                  // Stop tone
  }
  digitalWrite(LED, LOW);          // Turn off LED when done
}
```

Using a Dynamic Speaker

Piezo speakers aren't known for their musical fidelity. Most are engineered to produce fairly high frequencies, making the lower tones quiet or even indistinguishable. A dynamic speaker — the kind used in your home stereo (this speaker has a magnet and paper or plastic sound-making cone) — produces a wider range of frequencies. Most dynamic speakers have a low impedance, however, typically four to 16 ohms. The lower impedance can cause the speaker to draw excessive current from the Arduino, possibly damaging it.

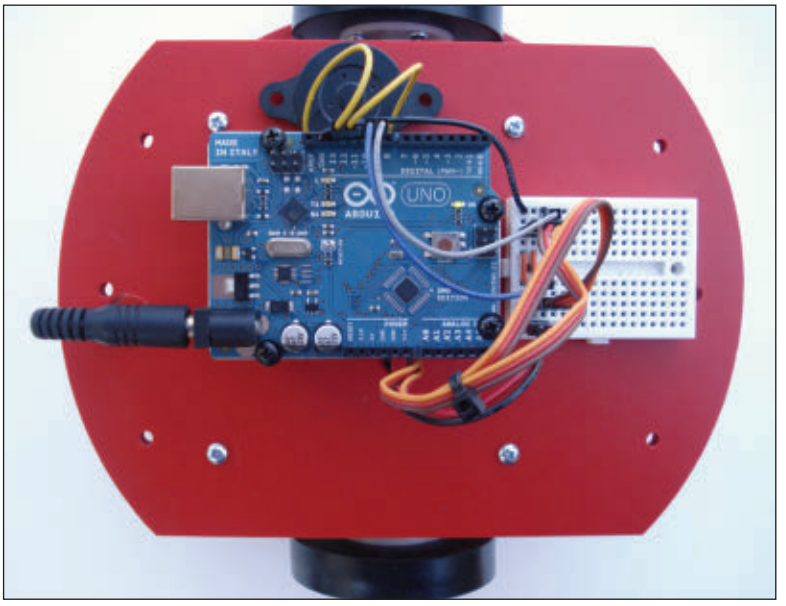


FIGURE 3. Use a small piece of double-sided foam tape to attach the piezo element to your ArdBot or other robot.

Arduino's I/O pins. Sound output may be diminished.

- *Use a speaker with a higher impedance.* Anything over 80 to 100 ohms should be sufficient. High impedance dynamic speakers aren't common, but they're not impossible to find. For example, Pololu sells a 100Ω 30 mm diameter speaker (item 1261).

Other methods involve using transistors, op-amps, or other circuits to act as a current buffer between the Arduino and the speaker. Though these simple circuits are easy to implement, they don't boost the sound. Dynamic speakers often require more drive to produce a loud tone, and for this you need an amplifier.

There are two common work-arounds to this dilemma:

- *Connect the speaker to the Arduino through a 100 ohm resistor.* This decreases the current to the speaker, making the connection safer for the

Adding an Audio Amplifier

The basic Arduino-to-speaker connection can deliver only so much sound power. Sometimes you must pump it up, and for this you need an audio amp. You can build your own amplifier from scratch, construct it from a kit, or purchase one ready-made.

A quick and easy option is the LM386 integrated amplifier IC. The sound output won't shatter any windows, but the chip is easy to get, cheap, and can be wired quickly.

FIGURE 4. Typical circuit schematics for the LM386 audio amplifier IC, showing 200:1 and 20:1 gain amplification. For louder sound, operate the chip at a voltage higher than the input. Be sure the capacitors you use are rated for at least twice the amplifier voltage.

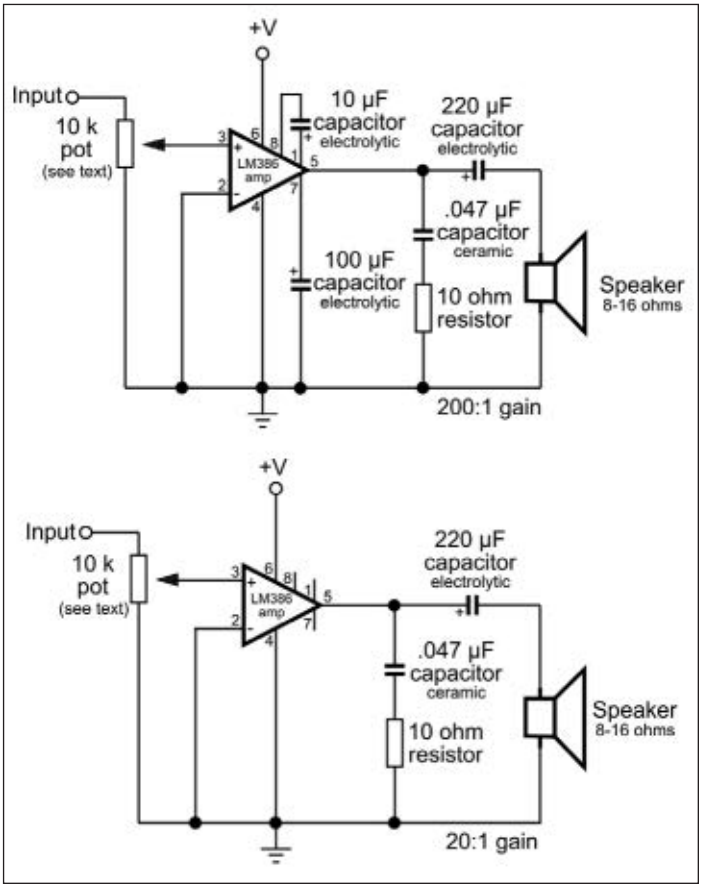
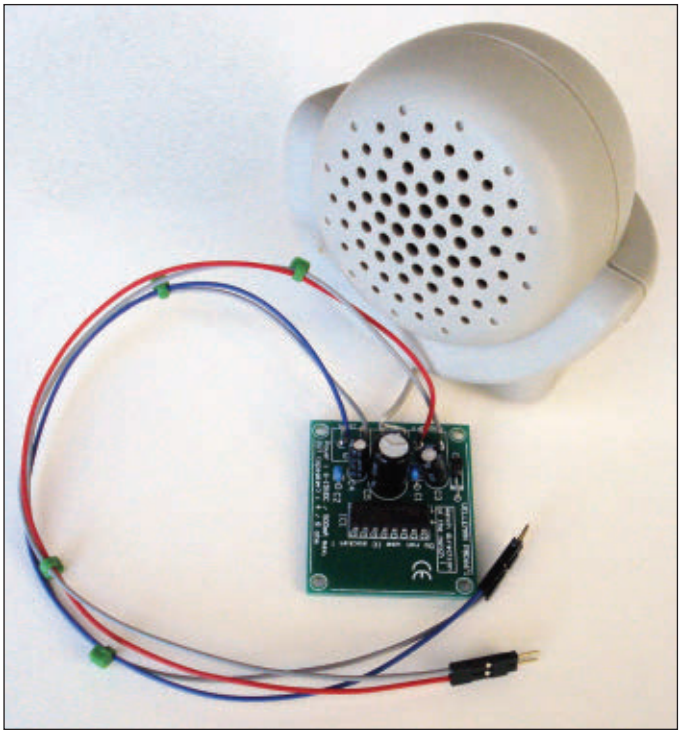


FIGURE 5. The Velleman three watt audio amplifier kit, shown constructed, attached to a computer speaker. Velcro or double-sided foam tape can be used to secure the speaker to your robot.



It's perfect for experimenting with sound projects. It has a *gain* — the ratio of sound in to sound out — of 20:1. By adding a few more components, the gain of the amp can be boosted 10 times to 200:1. Either amplifier will drive a small 4Ω to 8Ω speaker. See **Figure 4** for a schematic for both versions.

Here's a tip: For best results, make sure the 10 kΩ potentiometer is the *logarithmic* (or *audio*) *taper* kind, not the usual linear taper. When using a linear taper pot, the sound volume will only be affected at one end of the dial. With a logarithmic potentiometer, the volume change will be more evenly spread across the dial.

Personally, I prefer using amplifiers that someone else has built, preferably on printed circuit boards. They tend to be more reliable and sound better. Options include amplifier kits, hacked amps pulled from old audio and computer gear, and ready-to-go commercial models that you simply plug in.

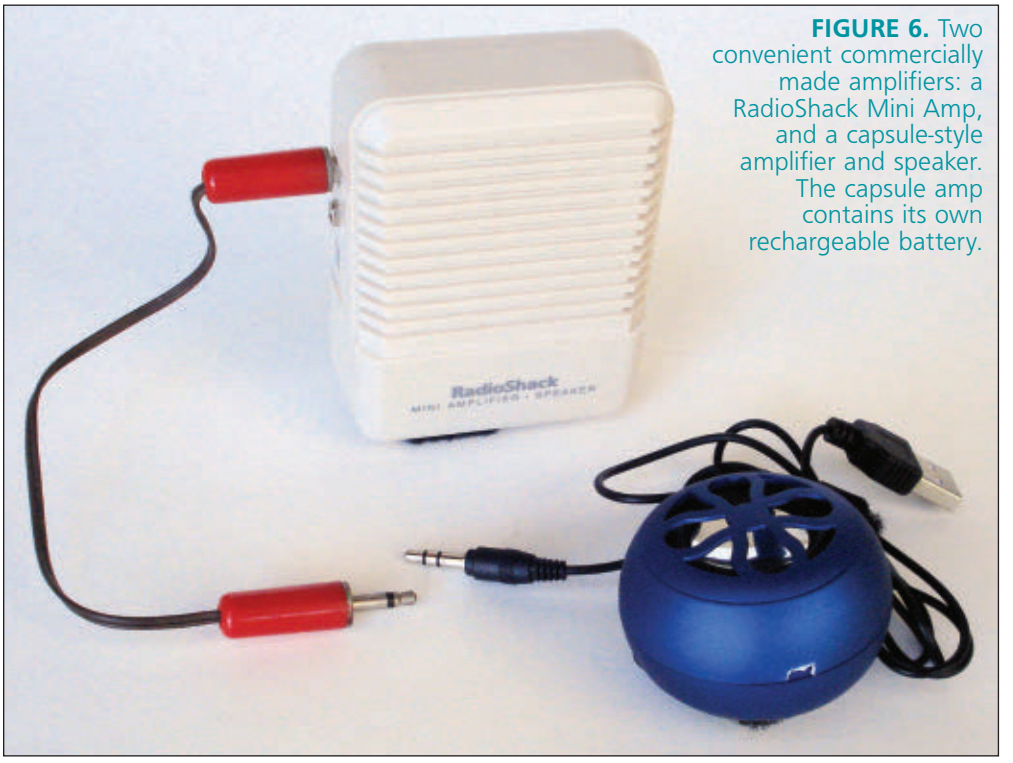


FIGURE 6. Two convenient commercially made amplifiers: a RadioShack Mini Amp, and a capsule-style amplifier and speaker. The capsule amp contains its own rechargeable battery.

Figure 5 shows a completed Velleman three watt amplifier kit, connected to a computer speaker I found at a flea market. I soldered plug-in leads to connect to both power (6-18 volts) and audio input. Sound quality is more than decent and loud enough to need a volume control. The assembly instructions for the amplifier show how to add one.

Velleman offers a seven watt version for even more output. With the increase in power, you need a more robust speaker. Make sure it's rated for the wattage of the amplifier if you intend to crank up the volume to 11. Otherwise, the speaker may become damaged.

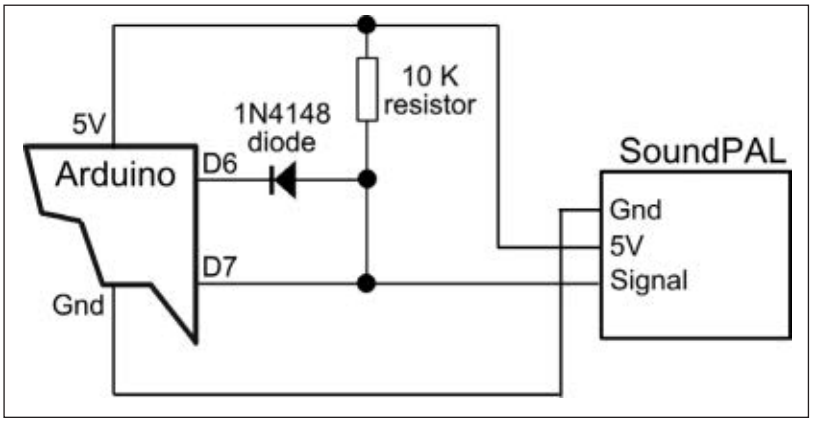
Powered computer speakers are a good source of amps to hack for your robotics projects. They're commonly found at garage sales and thrift stores. Sound quality (and loudness) is always better when the speaker is contained in its enclosure, so when possible keep the speaker in its box. If the enclosure is too large or adds too much weight to your robot, you can pull it out and mount the thing separately.

Commercial amplifiers are the easiest to use. They hold their own self-contained power source, so they don't need to connect to the Arduino's supply. One of my favorites for use in small robots is the "capsule" combination amplifier and speaker. They have their own

FIGURE 7. The Parallax SoundPAL is a totally independent sound co-processor and includes its own speaker element. The clear tubing is a clever "resonator" that greatly beefs up the sound level.

Photo courtesy Parallax, Inc.

FIGURE 8. The SoundPAL requires some external circuitry to connect to the Arduino. The diode and resistor allow the single I/O pin on the SoundPAL to attach to separate transmit and receive pins on the Arduino.



LISTING 3 – SoundPal.ino

```
#include <SoftwareSerial.h>

#define txPin         6         // Cathode end of diode
#define rxPin         7         // Anode end of diode
#define ledPin        13

#define SND_PLAY      0x01
#define SND_REPEAT    0x02      // Repeat count (1-254; 255 = infinite)
#define SND_AGAIN     0x03      // End the repeat block.

#define CHARGE        0x40      // Charge!
#define TAPS          0x44      // Taps
#define REVEILLE      0x5D      // Reveille
#define FIRSTPOST     0x7D      // Horse race bugle call
#define INTRO         0x8D      // Doo-doot doo doot doot DOOT
#define NYAH          0x93      // Nyah nyah nyah nyah NYAH nyah!
#define DEAD          0x97      // Funeral dirge
#define BATTHYMN      0x9D      // Battle Hymn of the Republic
#define DIXIE         0xA5      // Dixie
#define CUCARACHA     0xAC      // La Cucaracha
#define POPWEASEL     0xAF      // Pop! Goes the Weasel
#define MARSELL       0xB3      // Marsellaise
#define RULEBRIT      0xB9      // Rule Brittania
#define MATILDA       0xC0      // Walzing Matilda
#define KRADOUCHA     0xC6      // Kradoutcha ("There's a place in France")
#define WEDDING       0xCD      // Wedding March
#define ODE2JOY       0xD2      // Ode to Joy
#define DUDU          0xDA      // Du, Du Liegst Mir im Herzen
#define NOTIME        0xE1      // Rude sound
#define UHOH          0xE5      // Uh oh!
#define SIREN         0xE8      // American siren, infinite loop: reset
exits
#define PHONE         0xEE      // Rings once
#define WHISTLLE      0xF3      // Wolf whistle
#define CRICKET       0xFA      // Cricket

SoftwareSerial soundPal = SoftwareSerial(rxPin, txPin);

void setup() {

  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);

  playPresetNoWait(DIXIE, 1);      // Play Dixie, repeat 1 time
  delay(500);

  resetPal();
  delay(1000);

  playPresetWait(CUCARACHA, 2);    // Play La Cucaracha, repeat 2 times
  playPresetWait(DEAD, 1);

  playPresetNoWait(DIXIE, 1);      // Play Dixie again, without waiting
  soundPal.begin(9600);            // Open SoundPAL comm
  do {
    soundPal.print("?");           // Query SoundPAL
    delay(2);
  } while(soundPal.read() != 255); //Done when value is 255
  soundPal.end();                  // Close SoundPAL comm
  Serial.println("Done");

}

 void loop() { }

// Play preset tune; return from function
//  without waiting for tune to end
void playPresetNoWait(byte tune, byte repeats) {
  soundPal.begin(9600);                    // Open SoundPAL comm
  delay(10);                               // Brief delay
  soundPal.print("=");                     // Get attention of SoundPAL
  soundPal.write(byte(SND_REPEAT));        // Start repeat block
```

rechargeable (usually lithium-polymer) battery. Another favorite is the RadioShack portable amplifier, shown with a capsule amp/speaker in **Figure 6**. Sound may be channeled through the built-in speaker or routed to an external speaker.

Using the Parallax SoundPAL

While the Arduino is a perfectly capable sound maker, you may prefer a pre-fab solution where tones — even short songs — are made by sending a handful of simple commands. These *sound co-processors* do all the work of generating the PWM signals, freeing the Arduino to do other work — like operating servos and reading sensors.

One such co-processor is the Parallax SoundPAL — a tiny board with a big sound. **Figure 7** shows the SoundPAL which is billed as a "miniature sound player." Don't let its small size fool you; this thing puts out a lot of volume. **Figure 8** is the interface you need to connect it to the Arduino.

To make noise, you send the SoundPAL a combination of byte commands via serial communication. These commands define the frequency and duration of tones. You can even specify any of 24 built-in songs.

Additional commands let you repeat tones and songs any number of times, even indefinitely. Once commands are sent to the SoundPAL — which takes just a fraction of a second — the Arduino is free to turn its attention elsewhere.

Listing 3 shows a basic sketch detailing playing a sequence of the SoundPAL's

built-in repertoire of songs. I've defined functions to allow you to easily alter the sketch and use it as a framework for your robotic endeavors. A couple of things to note:

- The *resetPal* function immediately stops all sound output. This function is handy if the SoundPAL is in the middle of a long recital, or you've set the tones to repeat indefinitely.
- The *playPreset* function is provided in two forms: one where the function immediately returns after issuing the command string, and one where the function waits until the command has completed. The *playPresetWait* function demonstrates sending a query to the SoundPAL and checking the return value. When the value is 255, it means the SoundPAL has completed its current task.

Giving Your Robot a Voice

Not long ago, integrated circuits for the reproduction of human-sounding speech were fairly common. Several companies mass-produced these chips for voice-driven products like the Speak-and-Spell toys. Many of these ICs created unlimited speech, because they reproduced the fundamental sounds of the human vocal tract.

With the proliferation of digitized recorded voice — plus software-only speech built into Windows, Macintosh, and other computer operating systems — unlimited vocabulary hardware synthesizers have become somewhat rare. While most of the big chip makers have long exited the speech chip market, there are a couple of low cost solutions available to robot

LISTING 3 – SoundPal.ino continued

```
  soundPal.write(repeats);            // Specify # to repeat
  soundPal.write(byte(SND_PLAY));     // Cmd to to play preset
  soundPal.write(tune);               // Song to play
  soundPal.write(byte(SND_AGAIN));    // End repeat block
  soundPal.write(byte(0));            // Sequence end delimiter
  soundPal.print("!");                // Play sequence in RAM
  soundPal.end();                     // Close SoundPal comm
}

// Play preset tune; return from function after tune ends
void playPresetWait(byte tune, byte repeats) {
  digitalWrite(ledPin, HIGH);
  soundPal.begin(9600);
  delay(10);
  soundPal.print("=");
  soundPal.write(byte(SND_REPEAT));
  soundPal.write(repeats);
  soundPal.write(byte(SND_PLAY));
  soundPal.write(tune);
  soundPal.write(byte(SND_AGAIN));
  soundPal.write(byte(0));
  soundPal.print("!");
  delay(10);
  do {
    soundPal.print("?");
    delay(2);
  } while(soundPal.read() != 255);
  Serial.println("Done");
  soundPal.end();
  digitalWrite(ledPin, LOW);
}

// Reset SoundPAL (stop all sound)
void resetPal() {
  soundPal.begin(1200);     // Send min. 7.5ms LOW pulse to reset
  soundPal.write(byte(0));
  soundPal.end();
}
```

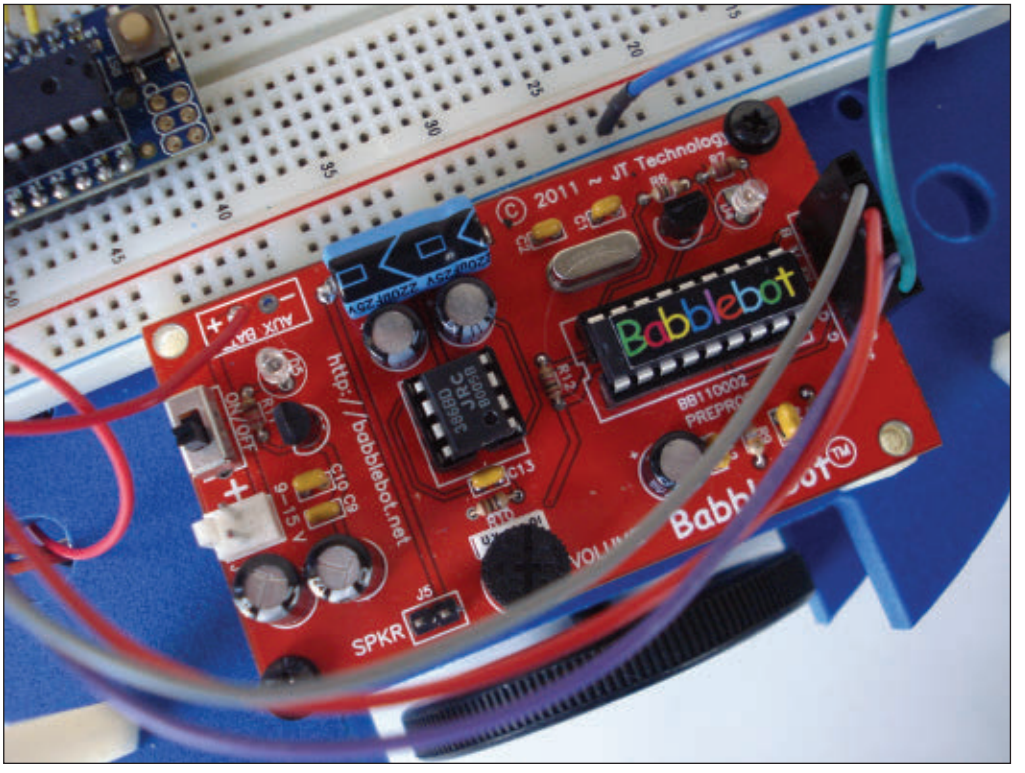


FIGURE 9. Babblebot mounted atop an ArdBot. Use long standoffs so you can fit a small speaker between the robot's deck and the Babblebot board.

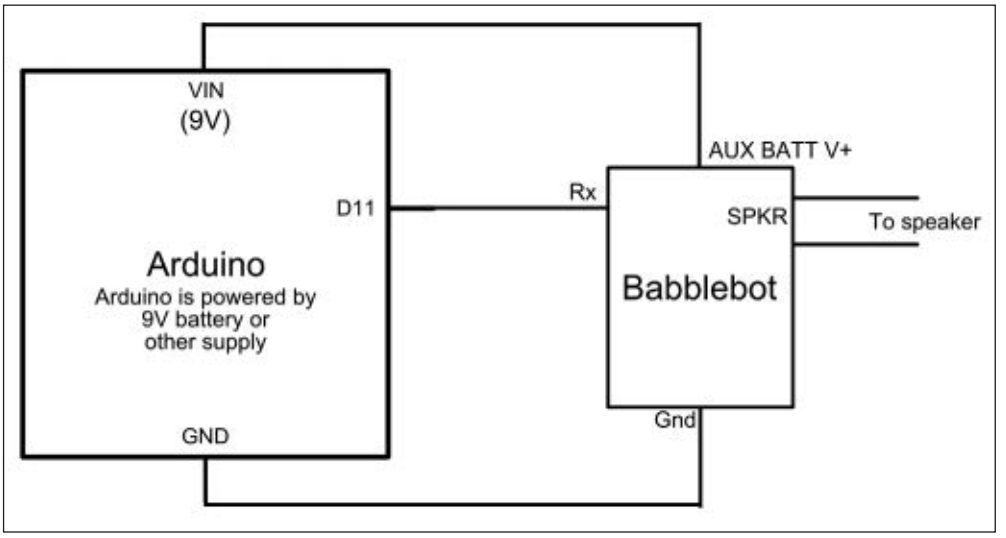


FIGURE 10. Electrical connection for the Arduino-Babblebot link. Be sure to power the Arduino from a 9V supply, and to connect the Arduino's VIN pin — NOT the 5V pin — to the Babblebot. This provides nine volts to the Babblebot.

LISTING 4 - SoundEqualizeDemo

Listing 4 — Babblebot.ino

```
#include <SoftwareSerial.h>
#include <SimpleTimer.h>     // Put into sketchbook libraries folder

#define Rx       255         // No pin (receive not used)
#define Tx       11          // Pin D11
#define mixerA   0
#define mixerB   1

SoftwareSerial babble(Rx,Tx);
SimpleTimer  timer;

enum soundginPreset {
  SpaceWarp          = 0,
  Waba               = 1,
  RandomThoughts     = 2,
  Gong               = 3,
  Pwang              = 4,
  Wow                = 5,
  Rananana           = 6,
  Twarty             = 7,
  Telly              = 8,
  Pulsator           = 9,
  Bound              = 10,
  TipToe             = 11,
  Spoles             = 12,
  Chopper            = 13,
  Phazer             = 14,
  PowerLines         = 15,
  HeavyMetal1        = 16,
  HeavyMetal2        = 17,
  ACMotor            = 18,
  YaYa               = 19,
  March              = 20,
  NoiseChatter       = 21,
  BlipChatter        = 22,
  Carney             = 23,
  EarthQuake         = 24,
  MindProbe          = 25,
  Siren              = 26,
  Squaba             = 27,
  SteamLoco          = 28,
  FreqMod            = 29,
  AmpMod             = 30
};


void setup() {
```

builders. These include:

- *SpeakJet*, from Magnevation
- *Babblebot* (used to be called the Soundgin), from JT Technology

Both ICs create human voice effects using allophones — discrete sounds made by the mouth and vocal chords during speech. The allophones are essentially strung together to create what sounds like someone talking. Though not as clear as a real human voice, the staccato nature of the sound, in fact, adds to the robotic effect.

Both the SpeakJet and Babblebot are complex sound generators, including "biologic" effects, TouchTone phone tones, and alarm sounds. The chips contain independent oscillators that can recreate the human vocal tract and sound effects, and in the case of the Babblebot, can also synthesize music.

As with the SoundPAL, simple serial communications connect either speech chip to a microcontroller. The SpeakJet is fairly well established with numerous examples on the Web, so I'll skip ahead and discuss the Babblebot in more depth.

Programming the Babblebot to Make Sound Effects

In addition to providing speech, the Babblebot makes sound effects and polyphonic (more than one note at a time) music. The Babblebot sound generator is available as a stand-

alone IC, as well as both an Arduino shield and a separate breakout board. The shield and breakout board contain a small audio amplifier. **Figure 9** shows the Babblebot breakout board mounted to an ArdBot.

Electrical connection is simple, and is shown in **Figure 10**. Power for the Babblebot is derived from the ArdBot's nine-volt battery. The Babblebot board contains its own regulator for the Babblebot logic chip, and the onboard LM386 audio amplifier uses the higher voltage to produce a louder sound.

Important! You must power the Babblebot at 9V to 18V. When connecting to the Arduino, plug in a 9V battery or power supply. Don't rely on just the USB cable. The five volts from the USB connection is not enough to power the Babblebot chip through its voltage regulator. When both the USB cable and external power are connected to the Arduino Uno, the board will automatically switch over to external power.

The Babblebot board is mounted on nylon standoffs. I've picked standoffs with a minimum length of 5/8" to provide enough room underneath for a 1" to 2" dynamic speaker. Generally speaking, the larger the speaker, the bigger the sound. The speaker should have a 4Ω to 16Ω impedance.

Listing 4 shows how to exercise the Babblebot's preset sounds; these include Telly (telephone ring), SpaceWarp, and Siren. The sketch uses a third-party Arduino timer library — *SimpleTimer* — to provide time delays for playing and then clearing the presets. The timer makes it unnecessary to use the Arduino *delay* statement which causes the entire sketch to halt whenever it's used. Even so, I've demonstrated *SimpleTimer* and the *delay* methods.

In order to use the sketch, you must copy the *SimpleTimer* folder to your Arduino sketchbook *libraries* directory. You can get the *SimpleTimer* files at **arduino.cc/playground/Code/SimpleTimer**. The necessary files are also included in the download of all the program listings for this article at the article download link.

The Babblebot contains two sound mixers (actually it has three, but the third is to combine the first two into a single output). These two mixers can make sound independently. That

LISTING 4 – SoundEqualizeDemo continued

```
  // Be sure Babblebot is already powered
  babble.begin(9600);
  delay(500);               // Keep at 500 or higher for
Babblebot timing

  // Load a preset into a mixer, and wait (using delay)
  loadPreset(mixerB, BlipChatter);
  delay(2000);              // Play for 2 seconds
  clearMixer(mixerB);

  // Load a preset unto mixerA, and allow timer to clear sound
  //   After this the loop function starts
  playSound(YaYa, 2000);    //Play YaYa for 2 seconds
}

// Main loop
void loop() {
  timer.run();              // Check if timer has elapsed
}

// Play a preset using timer
//   Set timer delay, and specify function for clearing sound
void playSound(byte preset, int delayInterval) {
  loadPreset(mixerA, preset);
  timer.setTimeout(delayInterval, clearSound);
}

// Clear sound, mixerA hardcoded (use with timer)
void clearSound() {
  clearMixer(mixerA);
}

// Clear sound, specify mixer (overloaded function)
void clearSound(byte mixer) {
  clearMixer(mixer);
}

// Load a preset, specify mixer and preset
void loadPreset(byte mixer, byte presetSound) {
  babble.write((byte)0x1b);
  if(mixer == mixerA)
    babble.write((byte)0x4b);      // Load preset in mixer A
  else
    babble.write((byte)0x6b);      // Load preset in mixer B
  babble.write(presetSound);
  triggerMixer(mixer);
}

// Trigger (play) a preset
void triggerMixer(byte mixer) {
  babble.write((byte)0x1b);
  if(mixer == mixerA)
    babble.write((byte)0x53);      // Trigger mixer A
  else
    babble.write((byte)0x73);      // Trigger mixer B

}

void clearMixer(byte mixer) {
  babble.write((byte)0x1b);
  if(mixer == mixerA)
    babble.write((byte)0x57);      // Clear mixer A
  else
    babble.write((byte)0x77);      // Clear mixer B
}
```

means you can play two presets simultaneously by specifying whether the sound is piped through *mixerA* or *mixerB*. Presets play indefinitely. When you've had enough of the sound, simply clear the mixer that's playing it.

The Babblebot sketch shows two methods of playing a preset:

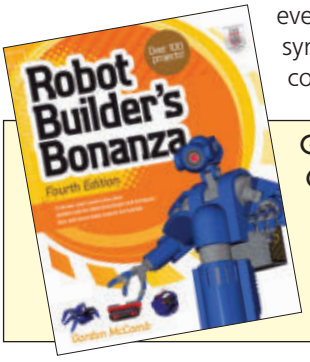
- Play by specifying the preset and mixer. You then use the Arduino *delay* statement to wait a given period of time before clearing (stopping) the sound.
- Play by specifying the preset and a duration. *MixerA* is used by default. When the duration period is over, a timer is triggered to clear the mixer which stops the sound.

Going further, you can program the Babblebot to play your own sound effects and songs; even to speak. The command syntax for these procedures is fairly complex; read the documentation for the chip at the **babblebot.net** website. (The documentation is for the Soundgin chip, but the only change is in the name. Internally, the Soundgin and Babblebot are the same.)

A better method of programming the Babblebot is to use a predefined library devised by the makers of the GinSing BabbleShield — a Babblebot add-on shield for the Arduino. Check it out at **ginsingsound.com**. This open source library greatly simplifies using the Babblebot. As with many open source projects, the GinSing library is routinely updated, so be sure to fully read any accompanying documentation for the latest changes. GinSing version 3.0 is updated for Arduino 1.0.

Wall to Wall Sound

So ends this episode on robots sounding off with the Arduino. Next up: How to use the Arduino to play MP3 and other digital audio files, plus making music and unusual sound effects with MIDI. **SV**



Gordon McComb is the author of *Robot Builder's Bonanza*, now in its fourth edition. Greatly expanded and updated, this best selling book covers the latest trends in amateur robotics, and comes with 10 all new robot construction projects, plus more ideas for building robots from found parts. Look for *Robot Builder's Bonanza, 4th Ed* in the *SERVO* Webstore at **http://store.servomagazine.com**. Gordon may be reached at rbb@robotoid.com.

TCF™-2012

The College of New Jersey Hosts
The 37th Anniversary of
THE ORIGINAL PERSONAL COMPUTER SHOW

37th Annual

Trenton Computer Festival™

KEYNOTE SPEAKER:
Jeff Gomez,
President & CEO,
Starlight Runner
Entertainment
on
"From the Inner City to Pandora: The Power of Story in a Tech-Driven World"

DOOR PRIZES

INNOVATION POWERED BY COMPUTING

Admission:
Advance Tickets - $10.00
(Door Prize for Advance Ticketholders)
www.tcf-nj.org ***(deadline March 4)***

One-Day Ticket (at gate) - $10.00
Rain or Shine - Free Parking

@ The College of New Jersey, Ewing, NJ
Saturday, March 10, 2012
10 am - 5 pm

- **Apple/Mac/iOS Day**
  Full Day with Apple's Dave Marra, N3ODX
- **Windows 8 Presentation**
  by David Soll, KC2YJI

65+ Talks, Workshops, Tutorials, Demos and Special Events!
TALKS ON WINDOWS, MAC and LINUX.

- *Special low winter prices for vendor and hobbyist tables in the indoor Flea Market. (See web site.)*
- *Also, special room rates at the Element Hotel in Ewing Twp., $109/night plus tax. Reserve a room in the ITPC/ISEC/TCF block by Feb. 14 by calling (609) 671-0050 or (877) 353-6368.*

MULTIMEDIA, 3D VIDEO, PHOTOGRAPHY
HOME CONTROL, SECURITY and WIFI
Get an Amateur Radio License at TCF!
Ham Cram Session & Exam
Arduino Developer/User Tutorial/Workshop

IMPORTANT CHANGES TO THIS YEAR'S FESTIVAL

- The Festival will be in March this year.
- The Festival will be only **ONE DAY** this year, on Saturday, March 10.
- Outdoor Flea Market vendors will be INDOORS this year, because of the early date of the Festival.
- The IT Pro and ISEC conferences will be on Friday, March 9.

Integrated STEM Education Conference (ISEC)
Friday, March 9 – **Theme: Integrate by Design.**
Discussion and exploration of best practices in K-16 STEM education (extra cost registration).
Continuing education credits available!
Info: http://ewh.ieee.org/conf/stem/

TCF IT PROFESSIONAL CONFERENCE
Friday March 9, 2012, 9am —5pm
For Conference Info and Fees See
http://www.tcf-nj.org/pc

The optional TCF IT Pro and ISEC Conferences require add'l fees.

For additional TCF'12 info, directions and advance tickets:
www.tcf-nj.org

SPONSORS
The College of New Jersey (TCNJ)
New York Amateur Computer Club (NYACC)
IEEE Region 1
IEEE Princeton Central Jersey Section, and
the Joint Princeton Chapters of the ACM and
the IEEE Computer Society

The 2012 Trenton Computer Festival™ is produced by TCF, Inc., a not-for-profit 501(c)(3) educational organization.

2012P-rev-04a

Product Review

My Time With The TurtleBot

by Alan Federman

www.servomagazine.com/index.php?/magazine/article/march2012_Federman

Over 30 years ago, a group of hobbyists were introduced to a prototype of what later became the Apple II. Since 1980, I've been wondering: "Where is the personal robot equivalent of the Apple II?" While it isn't quite here yet, I believe the TurtleBot by Willow Garage is a strong candidate for the "Apple" of personal robots, and at a price point below $1,500 is an extremely affordable experimental platform suitable for prototyping many innovative domestic applications.

What is a TurtleBot? According to *www.turtlebot.com*: "TurtleBot is a low cost personal robot kit with open source software. With TurtleBot, you'll be able to build a robot that can drive around your house, see in 3D, and have enough horsepower to create exciting applications."

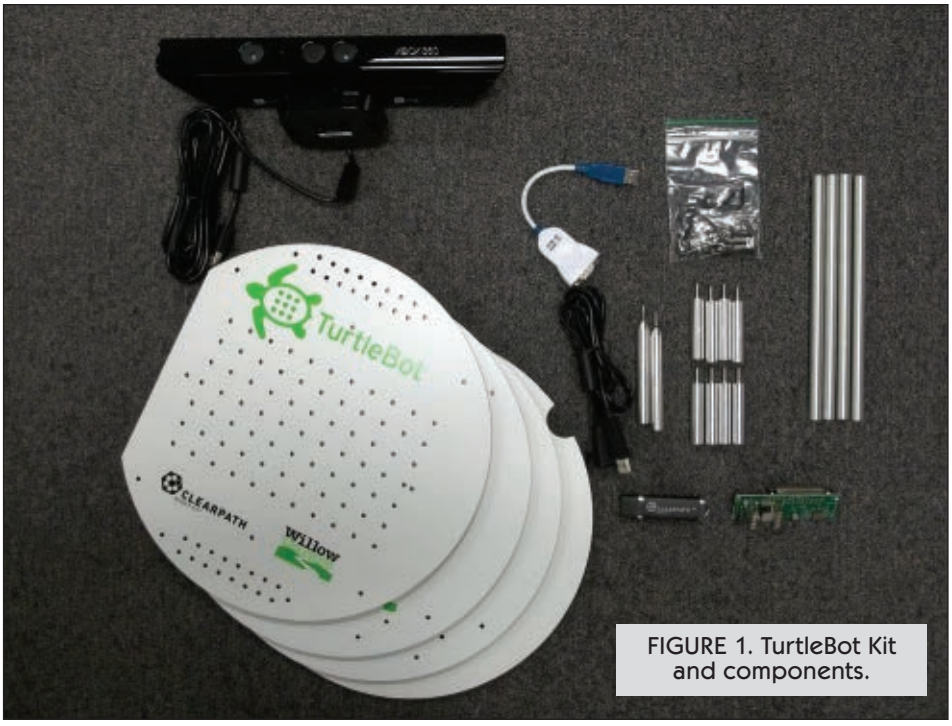


FIGURE 1. TurtleBot Kit and components.

The Homebrew Robotics Club (**http://hbrobotics.org;** HBRC) which meets in Mountain View, CA, is a direct descendant of the famous Homebrew Computer Club (HCC). If you haven't heard about the HCC, you might be familiar with some of its members like Steve Jobs, Steve Wozniak, Adam Osborne, and George Morrow. Over the last year, several members of the HBRC have been experimenting with the Willow Garage ROS (Robot Operating System) and Google's ADK (Android Development Kit) which is an Arduino based accessory controller that can be used in robotics. ROS and Arduino are now compatible.

In an effort to promote future development, HBRC was generously

given a complete TurtleBot assembled by Clearpath Robotics. The TurtleBot came with an Asus laptop running Ubuntu, an iRobot Create as the platform, and a Kinect as the 3D camera and range finder. The HBRC's plan is to loan the robot out on a monthly basis, and I was the first volunteer.

I configured it initially with another laptop (a dual boot Vista/Ubuntu using an AMD 64 processor). I used the 'workstation' as the control console to send commands over a local Wi-Fi network to the node running on the TurtleBot laptop. The HBRC's plan is to loan the robot out on a monthly basis, and I was the first volunteer.

Simultaneously — because I am a masochist and love to suffer — I sourced a Barnes and Noble Color Nook to Android Honeycomb 3.0. To my delight, this made a successful console into the TurtleBot. IMHO, the TurtleBot is a game changing introductory platform and at the price, can't be beat. Remember, the Apple II was just a toy until Visicalc came along.

My experience has made me thirsty for more. I am dreaming of an equivalent personal robot — a "TurtleBot II", if you will. A 25-50 lb platform big enough to open doors; autonomous enough to avoid running over the cat; and to be able to do light domestic work like cleaning, safety, and security; or even as an elder or disability aide.

Over the course of a month, lots of boneheaded mistakes, and numerous emails, the TurtleBot was returned to the club and successfully demonstrated. A good familiarity with Linux is very useful since the online tutorials assume you have this knowledge. Ubuntu has a great installer that can convert your old Windows laptop into a dual boot wonder machine. A caveat, however, is the learning curve for the TurtleBot is steep. To quote *Zen and the Art of Motorcycle Maintenance*: "Assembly of Japanese bicycle requires great peace of mind."

Assembly

It is possible to roll your own TurtleBot by buying the components separately. Fortunately, our kit was almost entirely assembled from Clearpath Robotics. Assembled or kit, be sure to print out the online manual and check off the steps. I spent a lot of time scratching my head because the power cable and adapter for the Kinect wasn't plugged in. Both Clearpath and Willow Garage provide great support and are receptive to suggestions. For example, users complained about chasing foam packing peanuts when opening the shipping box, so Clearpath no longer uses them.

Configure your workstation and turtle so they know which is what! In this case, my workstation was 192.168.2.149 and the Turtle ended in .146. It would probably be easier to do this as a VPN, but I configured the environment to work with the local Wi-Fi. The following commands set up the environment on the workstation.

FIGURE 2. Assembled TurtleBot.

Workstation Environment

On both computers, you need to set up the path and network address variables. In Linux, this can be done by editing the startup shell script which is named '.bashrc.' The instructions from the ROS Wiki are very good, but occasionally not completely up to date. Check out **www.ros.org/wiki/turtlebot_bringup/Tutorials/TurtleBot%20Bringup**, as well.

```
ifconfig -a                    <- shows network configuration if
                               <- wireless is available
ping 192.168.2.146             <- tests communications
ping 192.168.2.149

echo export ROS_MASTER_URI=http://192.
   168.2.146:11311 >> ~/.bashrc
echo export ROS_HOSTNAME=192.168.2.
   149 >> ~/.bashrc
sudo apt-get install chrony          <- sync time and
                                     <- date off
                                     <- network
date

ssh turtlebot@192.168.2.146    <- to login to the
                               <- "TurtleBot" from
                               <- the workstation
```

Updating/Installing ROS TurtleBot Software on the Workstation

I already had ROS Diamondback on my workstation for

my Arduino/Android robots. Most of these commands are entered via cut and paste from the ROS/TurtleBot Wiki Tutorials. In this case, getting the correct version of both ROS (Diamondback) and Ubuntu (Maverick) is important. The following CLI commands should get the job done:

```
sudo sh -c 'echo "deb http://packages.ros.org/
    ros/ubuntu maverick main" >
/etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O -
    | sudo apt-key add -
sudo apt-get update
sudo apt-get install ros-diamondback-
    turtle-desktop
sudo apt-get install ros-diamondback-
    turtlebot-desktop
```

Install Software On the TurtleBot Laptop

Get the latest instructions from **http://ros.org/wiki/Robots/TurtleBot/Robot%20Setup**.

Here is a summary:

```
sudo apt-get update

sudo apt-get install ssh ros-diamondback-
    turtlebot-robot

sudo ntpdate ntp.ubuntu.com

sudo apt-get install chrony <- Important for syncing
                            <- time and date
```

Configure TurtleBot Laptop Environment

```
    ifconfig    <-get the IP numbers
    echo export
ROS_MASTER_URI=http://192.168.2.146:
     11311 >> ~/.bashrc
    echo export ROS_HOSTNAME=192.168.2.146
     >> ~/.bashrc
```

Starting/Powering On the TurtleBot

The startup process on the TurtleBot is to turn on the laptop and login as 'turtlebot.' (This starts the network, though there is a way to autoboot to avoid this step – another exercise for the student.) Next, power on the Create, and start the turtlebot service:

```
sudo service turtlebot start
```

Once this is complete, everything else can be done from the workstation either directly or through an SSH terminal session.

Some actions like starting the camera and 'chirp' need to be launched via an SSH window; some will work off both computers (teleop), and rviz (more on this in a moment) and voice control come from the workstation. It depends on where the hardware that supports the processes is physically. For example, since the microphone is probably attached to the workstation, voice commands are launched from it. Chirp launched on the workstation will go to the workstation's speaker. Chirp launched via the SSH terminal will make the TurtleBot "moo."

Controlling the TurtleBot From the Workstation

I put these two commands into a single script called 'turtle' in my home directory. I started the dashboard by typing `./turtle`. (Note: the TurtleBot needs to be powered on and the TurtleBot service needs to be running before you can initiate the dashboard.)

```
source /opt/ros/diamondback/setup.bash
rosrun turtlebot_dashboard turtlebot_dashboard&
```

The dashboard program really is essential, as it monitors the battery status. The pull-down from the icon that looks like a gear sets the robot's mode. You set the robot "active" which enables the robot to move. The dashboard is needed to put the robot into passive mode so that it will charge the batteries correctly. It is also important to make sure breaker '0' is green, since this means power is going to the Kinect.

FIGURE 3. The TurtleBot Dashboard all green — indicating the robot is ready to roll.

Acknowledgements

I had a lot of help in getting the TurtleBot up and running. Firstly, to Wayne Gramlich and Ralph Gnauck of HBRC, who proposed the idea of a monthly checkout of a club TurtleBot. The staff at Willow Garage agreed to the proposal and made the robot available. Furthermore, they offered both email and person-to-person support. Included are Tully Foote, Mike Ferguson, Adam Stambler, Melonee Wise, and Ken Conely. Matt Rendall, Jared Lafler, and Ryan Gariepy of Clearpath Robotics had a couple of good tips and provided photos. Several members of HBRC also had good ideas and useful suggestions, and acted as a cheering section to keep me going on my one month sprint. Special thanks to Patrick Goebel whose Pi robot project was a bridge between "ROS and the Rest of Us." Thanks also to everyone who reviewed the manuscript. I don't understand the cosmic significance, but most of the HBRC members that helped me have last names that start with "G." I suspect a virus.

Starting the Kinect

If this command does not work, you will see diagnostic error messages. In that case, you may need to update the drivers.

```
:roslaunch turtlebot_bringup
 kinect.launch          <-starts Kinect if in
                        <-'full' mode
```

Fixing the Kinect by Updating the Drivers

After getting the robot to move, I had some difficulty operating the Kinect. At first, it was because of an assembly error, but after that was corrected, some software drivers needed to be updated:

```
rosrun dynamic_reconfigure
reconfigure_gui

sudo apt-get install ros-diamondback-
    openni-kinect
hg clone https://kforge.ros.org/openni/drivers
cd drivers
make
sudo apt-get install git
make
more ~/.bashrc
```

Starting RVIZ

The rviz program is the master visualization program that is used for the camera and LIDAR systems. It is the basis for determining the robot's location for SLAM (Simultaneous Location and Mapping).

```
rosrun rviz rviz
```

Installing Voice Command Operation

```
svn checkout http://albany-ros-pkg.google
    code.com/svn/trunk/rharmony rharmony
sudo apt-get install gstreamer0.10-pocketsphinx

Launch Voice Control

roslaunch pocketsphinx turtlebot_voice_cmd.launch
```

www.youtube.com/watch?v=eWhPVnOaD4k
Turtle Voice Control Teleop

Controlling the TurtleBot From an Android Device

If you have an Android device with Android Market, you can try downloading the free Willow Garage apps for controlling an ROS node. I've downloaded the "App Chooser" and "Teleop," and have gotten both to work on a sourced Barnes and Noble Color Nook.

FIGURE 4. Author demonstrating Android-TurtleBot Teleop 10/26/2011 at HBRC meeting, though usually liberal, I don't lean left that much normally.

www.youtube.com/watch?v=95oQk4BFABk
Android Tablet TurtleBot Teleop

Concerns and Conclusions

My main concern with the TurtleBot is battery limitations inherent in the Create design. The Kinect draws a lot of power, and it is difficult to determine the actual state of the batteries. Willow Garage notes: '... The Create will not initiate a recharge cycle if the battery is low, as it has already tried charging itself.' Probably the best way to operate it is to have two batteries and an independent charging station, and to swap out batteries every hour or so.

A Create weighs 6.7 lbs and an assembled TurtleBot is about twice that, so long autonomous run times are not possible. ROS and the TurtleBot are being developed under the open source philosophy; as a consequence, everything is evolving quickly, and documentation and tutorials are always in need of revision. Fortunately, there is a wide base of TurtleBot and ROS users, so assistance is just an email or **www.ros.org/wiki/ROS** Wiki away.

The significance of the TurtleBot can't be understated. You don't need a half million dollar research budget to begin doing state-of-the-art robotics development; $1,500 will get you plenty. **SV**

CNC From A Robot Builder's Perspective

by Michael Simpson

www.servomagazine.com/index.php?/magazine/article/march2012_Simpson

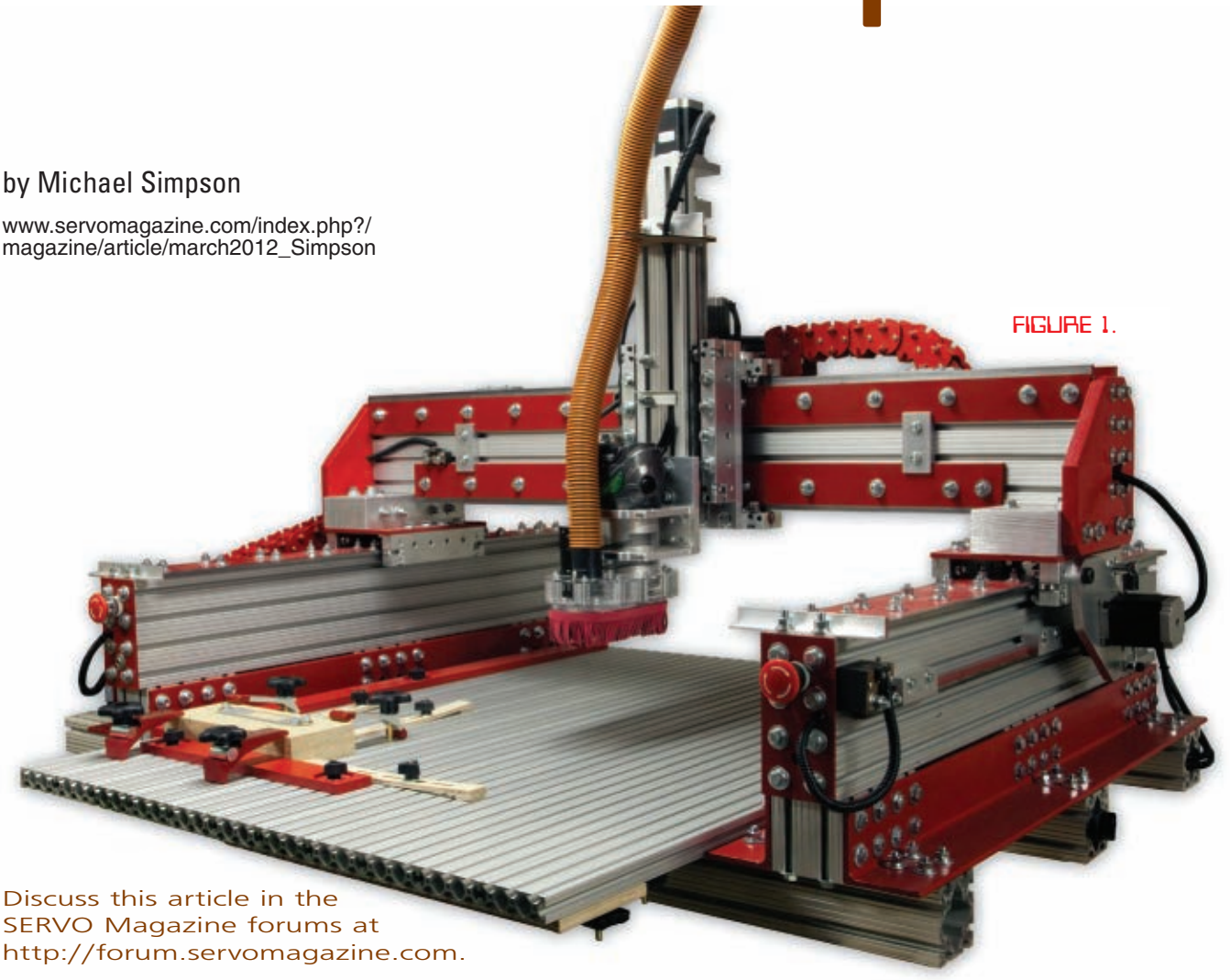


FIGURE 1.

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com.

Having built quite a few robots over the years using woodshop techniques and engineering best practices, I never dreamed that I would one day look back on those early days and wonder how I ever managed. After discovering CNC and its ability to accurately and consistently create parts from even the most complicated design, there was no going back.

My History with CNC

I got my first taste of CNC back in 2003, when I purchased a Sherline Mini Mill and a CNC conversion kit. I assembled the machine, added the conversion, and ended up using it for placing holes in various plastic enclosures. At that time, there was very little information available for the DIY CNC builder, so I never did much with the machine. It collected dust for a year or so, and I eventually sold it.

CNC Build 1

Years later, I started working on custom PC cases like the one shown in **Figure 2**. These cases were made by hand, and I would spend hours on the legs alone. A friend later asked me if I could build one of my cases for him, but

remembering the amount of work involved in creating something by hand made the prospect unappealing. I turned him down, but it started me thinking about ways to incorporate robotics into the design and machining equation, and I started research on a router-based CNC.

Thinking back to the difficulties I encountered researching projects for my first machine, I wasn't expecting much. However, I was astounded at the amount of CNC information available for the do-it-yourselfer. Books, websites, and forums of like-minded individuals all pointed to the fact that times had changed. It was time to build my first CNC router. I purchased books from Amazon and various plans from the Internet, and read several hundred posts on the various CNC forums. The first build — based on a book design — was made from MDF (medium density fiberboard), skate bearings, and threaded rod. I spent about $800 total.


FIGURE 2.

The most expensive part of this build was the electrics. The term "electrics" refers to the stepper motors, controller, power supply, cables, and sometimes the router. Out of the $800 I spent on this build, $650 went to high quality electrics which were reused on future builds. I purchased my electrics from a company called CNCRouterParts. They offer an electrics "kit" which is as close to plug and play that you can get, and I have used the same kit on all of my machines — large and small.

While this machine worked, it lacked consistency and accuracy, and the machining process often required more than one attempt. However, the build was not a complete failure. I discovered the following:

1. MDF is not a good material for the structural elements on a CNC. It is much too flexible.
2. Skate bearings are not suited for a CNC. They have too much play and do not wear well.
3. Threaded rod is not suited for a CNC. It has too much play, wears very badly, and is not efficient.

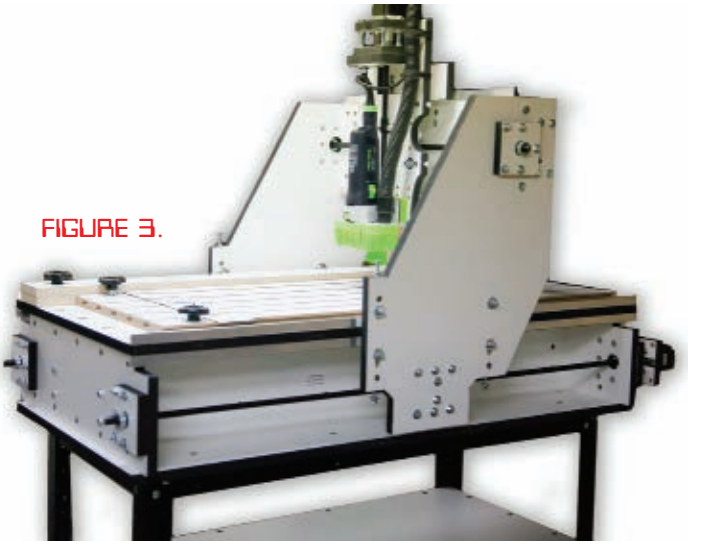


FIGURE 3.

With the knowledge gained from this build, I dismantled the machine and tried again with my own design.

CNC Builds 2 and 3

Taking lessons learned from my first attempt at building a CNC, I replaced the MDF with melamine-coated particle board, the skate bearings with V-bearings, and the threaded rod with a set of ACME lead screws and nuts. The resulting CNC machine (**Figure 3**) was a complete success. It consistently performed accurate cuts and I was able to build complex projects such as the clock shown in **Figure 4**.

Totally satisfied with this machine's performance, I decided to write a book showing the building process in a step-by-step format, complete with photos. However, since Build 2 was already complete, I had to build another machine in order to take pictures of the process. A video of the Build 2 machine is available on YouTube at **www.youtube.com/watch?v=BcJHmpaC_yc**.

When writing the book, I had some issues in conveying the build process through photography, so I decided to teach myself AutoCAD so I could create illustrations from any angle. With this new tool at my disposal, I designed the KRMx01 CNC – my fourth CNC build.

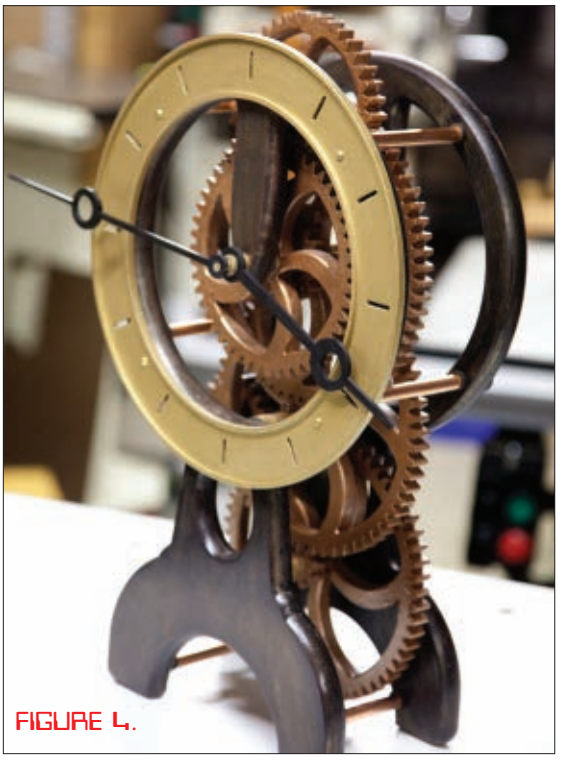


FIGURE 4.

CNC Build 4

The KRMx01 CNC (shown in **Figure 5)** is made from steel and aluminum. It has a cutting area of 43" x 32" in order to build larger projects. The new design of the KRMx01 also includes a lower gantry design for rigidity.

In addition to simplifying the overall design for the book I was writing, I added a few features that were part of the wish list I kept in my head while working with my previous CNC designs. As shown in **Figure 6**, these included particleboard shelves, an MDF T-Slot clamp table, an air exchanger, a dust shroud, and an E-chain for cable management.

This machine was a dream. I used it to construct parts for many projects, and even though I have built other machines since this model, it is still in operation. The step-by-step book based on the KRMx01 CNC took nearly six months to complete, but it was well worth the effort.

You can see Build 4 in action on YouTube at **www.youtube.com/watch?v=GR_nihzmlgI.**

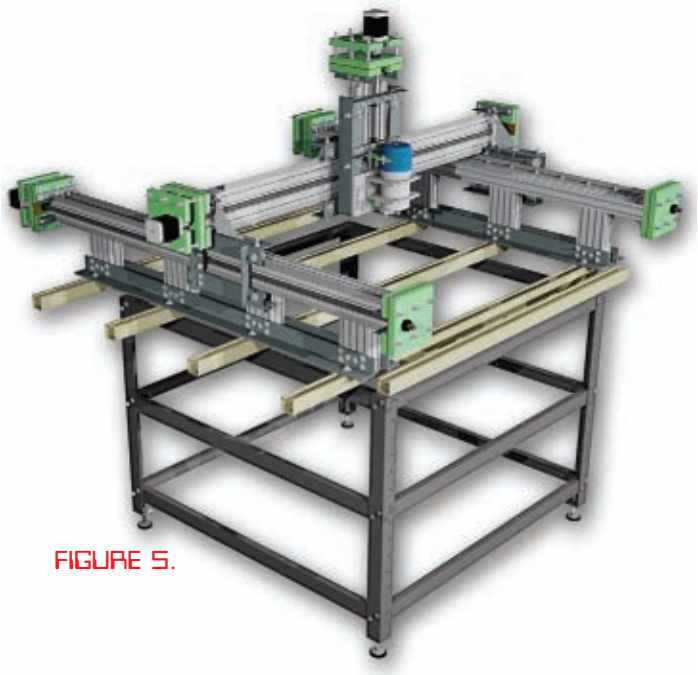


FIGURE 5.

CNC Build 5

While I considered the KRMx01 a success, a true DIY CNC builder never stops looking for ways to improve the process. We are also always looking for an excuse to build the next machine. As with many things, it isn't just about the destination. It's about the journey.

In early January 2012, I was asked to demonstrate my KRMx01 CNC at the Washington DC Science and Engineering fair in late April. Doing so brought some problems to the forefront. The machine wasn't designed to be transported, so it was too large to fit through my shop door. In addition, the design was not conducive to dismantling and reassembling. What to do? Design a new machine! And thus, the KRMx02 CNC was born.

While designing this new CNC machine, I wanted to incorporate the following features:

- Easy to build design adaptable for various size machines.
- Lower footprint to cutting area overhead.
- Industrial strength (super rigid).
- Upgradable.
- Removable gantry from CNC base.
- Rack and Pinion drive.

The KRMx02 (shown in **Figure 1** at the beginning of this article) fits the bill. It has a 30" x 27" cutting area and sports a full aluminum clamp table.

CNC Basics

Now that you know my history, I want to show you some of the robot related items I have built with my CNC machines. Before proceeding, let's review some CNC basics. This will give you a head start if you decide to build your own CNC.

CNC stands for Computer Numerical Control. It is a process where a computer is used to automate the machining process. There are numerous CNC style machines, but the one I am about to explain is a basic CNC mill, or router, in my case.

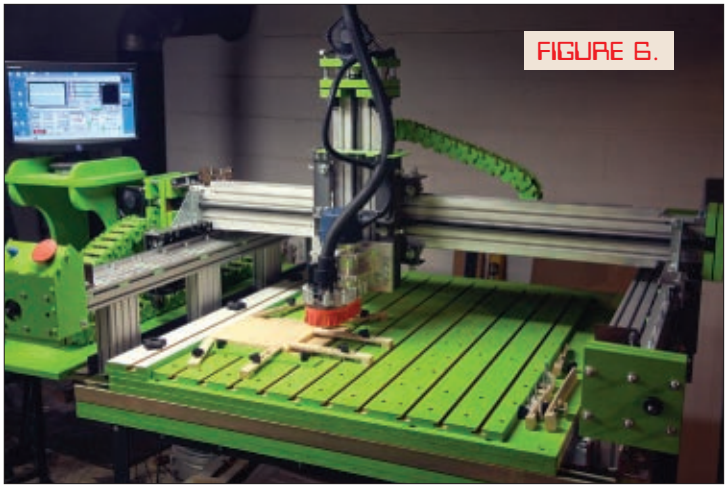


FIGURE 6.

Sources

KRMx01 Book
www.kronosrobotics.com/krmx01/index.shtml

KRMx02 Book
www.kronosrobotics.com/krmx02/index.shtml

CNCRouterParts
www.cncrouterparts.com

Sabertooth 2x25
www.dimensionengineering.com/Sabertooth2X25.htm

In addition to the two books mentioned above, the Kronos Robotics website also has projects, upgrades, and G-code downloads for CNC routers.

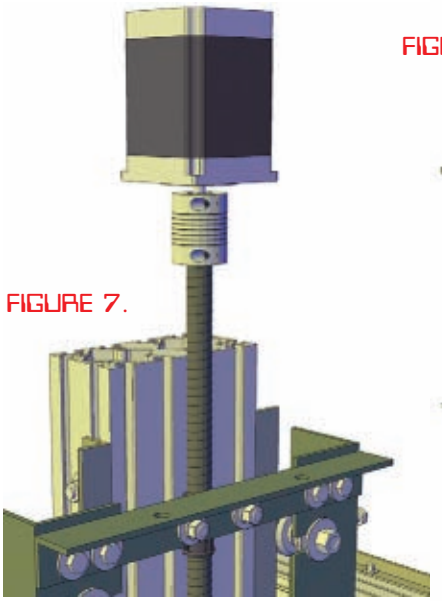


FIGURE 7.

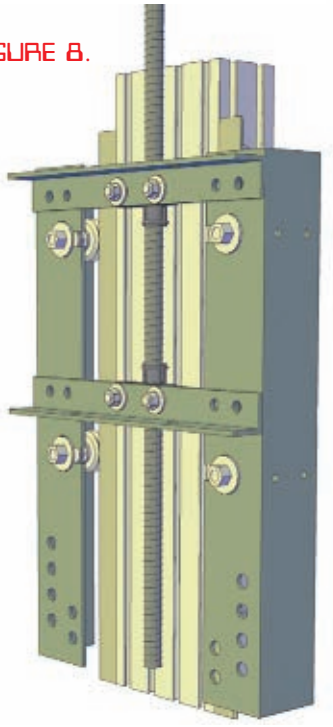


FIGURE 8.

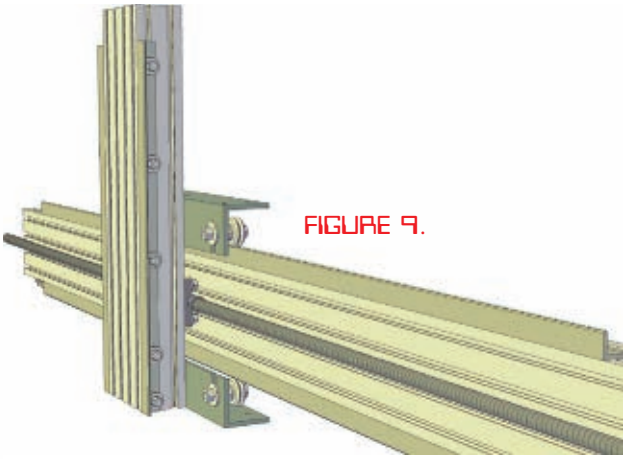


FIGURE 9.

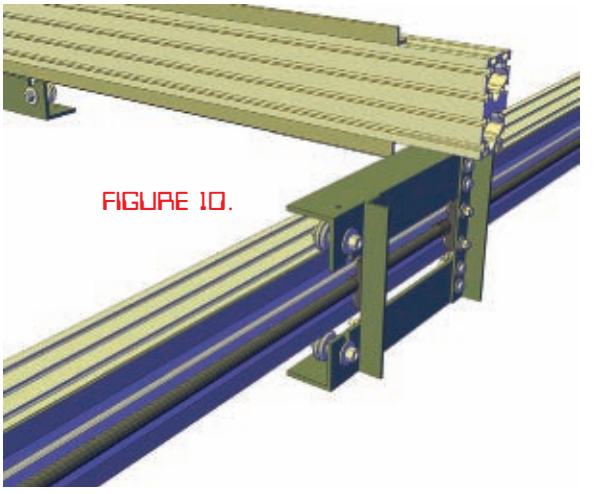


FIGURE 10.

Stepper Motors

Stepper motors are attached to the various ACME screws with couplers, as shown in **Figure 7**. The stepper motor receives instructions from a controller that is connected to your PC. The PC talks to the controller via controller software like Mach 3.

Z Carriage

The Z carriage (shown in **Figure 8**) moves up and down on a set of rails attached to the Z beam via an ACME nut that is connected to the carriage. An ACME screw is passed through the nut and as it turns, will raise or lower the carriage depending on the direction the attached stepper motor turns.

X Carriage

The Z beam is attached to the X carriage. The X carriage rides on a set of rails attached to the X beam, as shown in **Figure 9**. An ACME screw runs the length of the X axis passing through an ACME nut attached to the X carriage. As it turns, it moves the X axis to the left or right depending on the direction of the attached stepper motor.

Y Carriage

Each end of the X beam sits on a Y carriage as shown in **Figure 10**. The Y carriage rides on a set of rails attached to a Y beam. An ACME screw passes through an ACME nut attached to the Y carriage. As the ACME screw turns, it pulls the carriage forward or pushes it backward depending on the direction the attached stepper turns.

There are two Y beams — one on each side of the CNC. It is possible to drive the Y axis with a single ACME lead screw down the middle. This is not recommended due to the high loads the router can place on the CNC frame. When machining off center, racking can occur. For this reason, it is best to drive both carriages independently.

Rack and Pinion Drives

Another drive method used to move the carriages is the rack and pinion drive like the one shown in **Figure 11**.

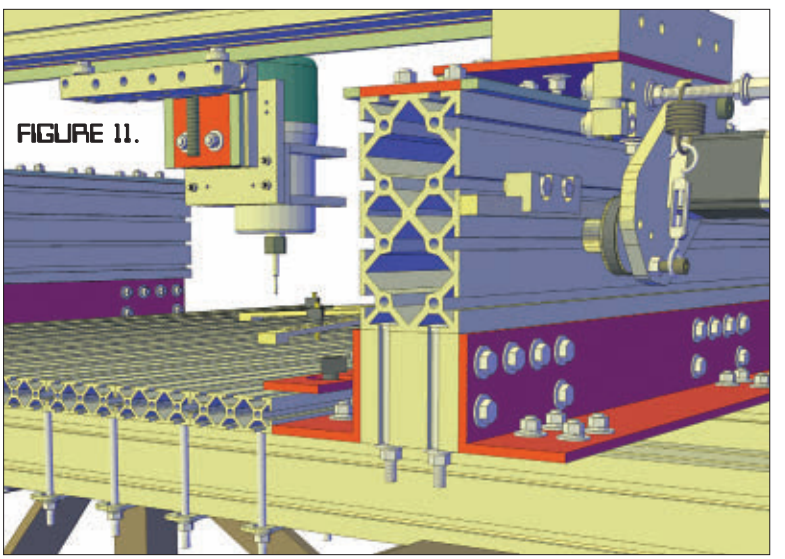


FIGURE 11.


FIGURE 12.

There are advantages and disadvantages to both drive methods. Additionally, there are also different bearing and rail methods. **Figures 8-10** are using a V bearing and rail system. **Figure 11** is using a flat rail and bearing system.

Ball Screw

There is a third drive system used to move the carriages. The ball screw works very similar to the ACME screw and nut. The nut on the ball screw has tiny ball bearings that fit into grooves in the threads of the shaft. This makes ball screws more efficient and able to handle higher loads, but they tend to be very expensive and are not practical on larger CNC routes.

Workflow

Getting your CNC to move requires a three-step process. First, you must design a part before you begin to program your CNC. Part designs are created using Computer Aided Design (CAD) software. CAD software allows you to create a drawing consisting of lines and curves. These lines and curves are then turned into a language called G-code by Computer Aided Manufacturing (CAM) software. This conversion process is not automatic. You tell the CAM software the diameter of the bit used and what side of the line you want it to cut. You also dictate the speed of the cut and the depth at which the bit is to cut.

G-code is used by the controller software to direct your CNC stepper motors on where to move, how fast to move, and through what path to move. Once G-Code has been created, controller software such as Mach 3 takes the information and sends pulses to the controller. The controller moves the stepper motors, and the CNC operation begins.

For the CAD software, I use Corel Draw which is relatively easy to learn. Another popular CAD program is AutoCAD. The CAM program I use is Cut-2D by a company called Vectric. They also offer advanced CAM software applications such as V-Carve Pro.

CNC Parts

There are various sources for the components used to build CNC machines. One company that I have dealt with a great deal is CNCRouterParts at **cncrouterparts.com**. They sell a great deal of components for the CNC do-it-yourselfer. As previously mentioned, the kit they sell is the best darn electrics kit available. The kit shown in **Figure 12** comes with four 12 foot cables, motors wired with connectors, a

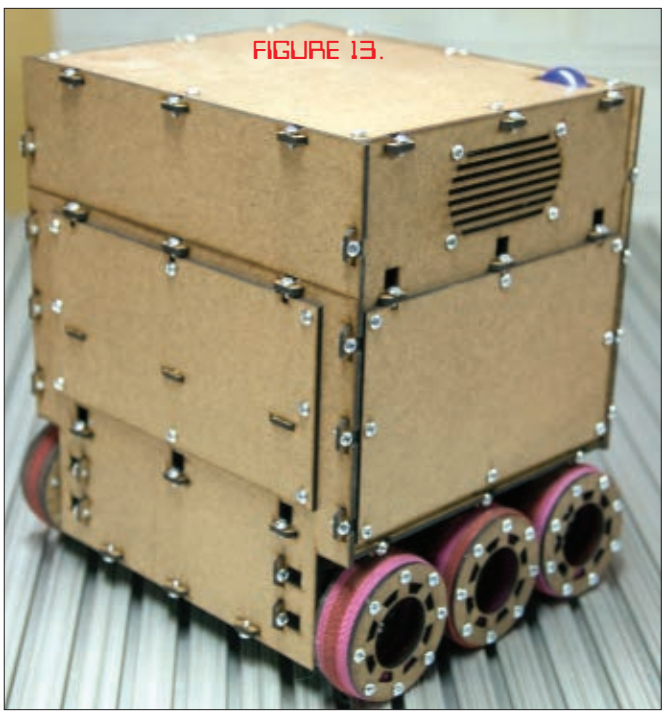


FIGURE 13.

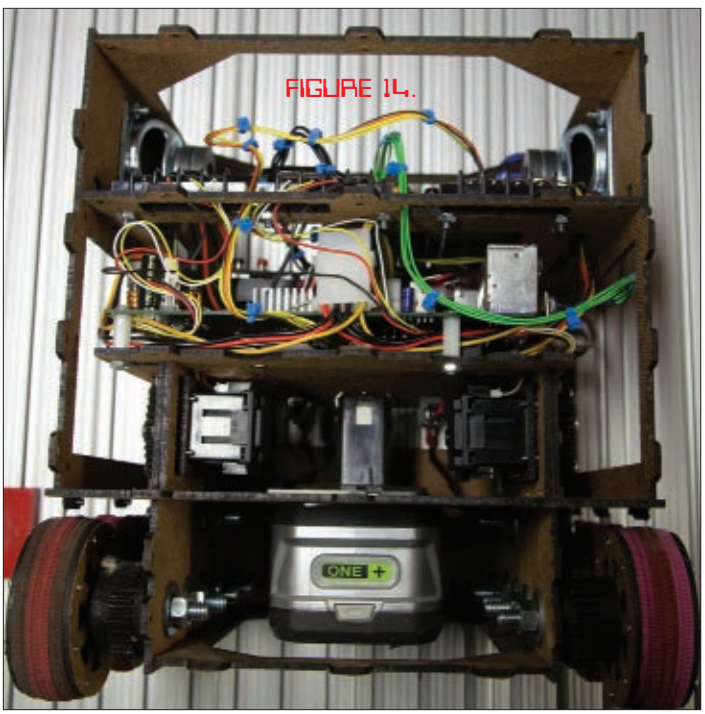


FIGURE 14.

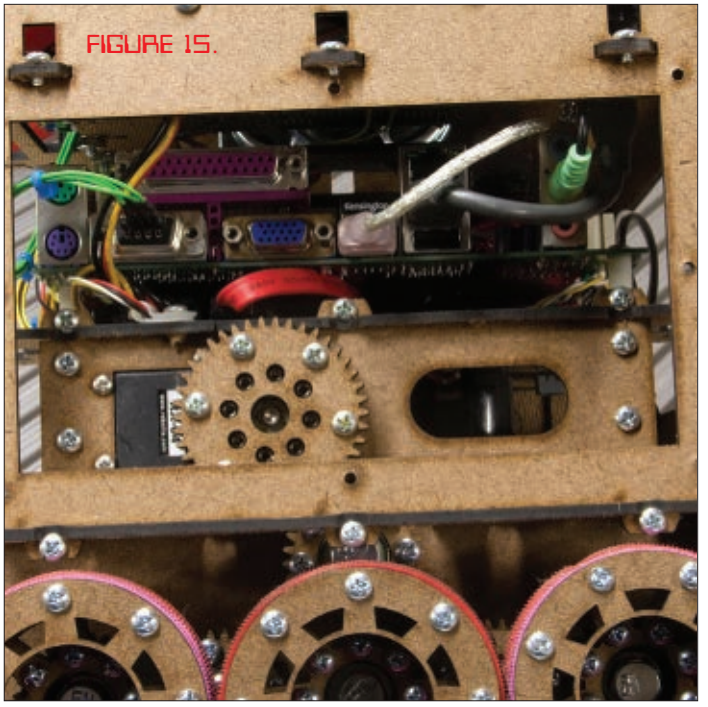


FIGURE 15.

12.5 amp power supply, and a G540 controller that is practically indestructible.

CNC Robot Projects

Building robots is fun. However, with a CNC at your disposal, new avenues of design possibilities emerge. The following are just some of the robots I have created with my KRMx01 CNC.

Frankenstein

Frankenstein (**Figure 13**) is a six wheeled robot driven by two RX-64 actuators. Frank features an ITX PC motherboard running Windows 7 for his brain and is programmed remotely via Wi-Fi. Every mechanical aspect of this robot was cut with the KRMx01 CNC. A 1/16 router bit was used to machine the 1/8" hardboard frame.

Frank features a leveled design that allows the use of a Ryobi One+ battery for power. The battery snaps into a cutout in a lower level (**Figure 14**), while the connectors at the level above make contact with the two battery terminals. All of the electronics are placed on the upper levels.The gears were cut from four layers of 1/8" hardboard. The main gear (shown in **Figure 15**) drives an idler gear that, in turn, drives the center wheel. Two additional idler gears connect the remaining wheels.

Want to see Frank in action? Check him out at **www.youtube.com/watch?v=mBnxZLw7tVA**.

Battle Bots

The battle bots shown in **Figure 16** were created for an upcoming Robot Fest. Attendees had a chance to control them as they faced each other in an arena. Six RX-64 actuators drive each robot.

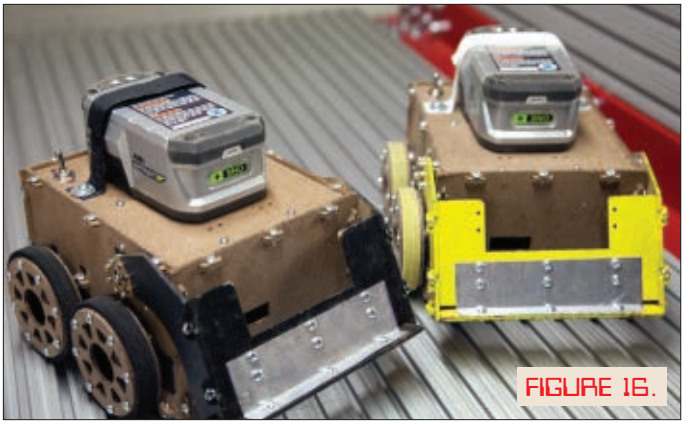


FIGURE 16.

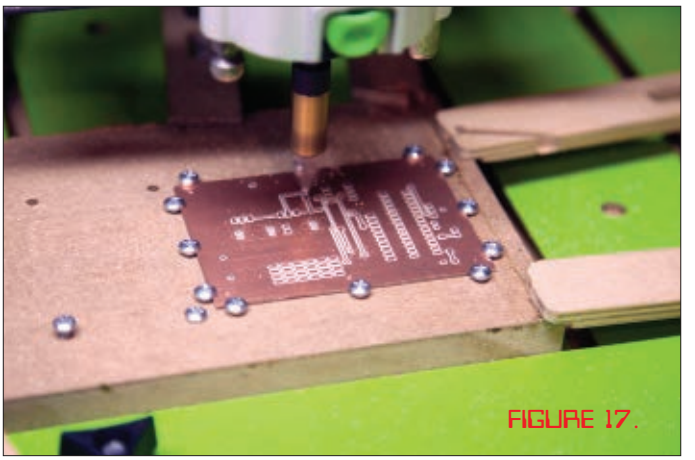


FIGURE 17.

Again, all the parts were machined on the KRMx01 CNC — even the main controller board shown in **Figure 17**. This is done by a process called isolation routing. The CNC removes material from a copper clad board to create the traces. Double-sided boards are no problem with the proper fixture.

The robot base for each battle bot consists of two layers. The lower layer shown in **Figure 18** holds the four

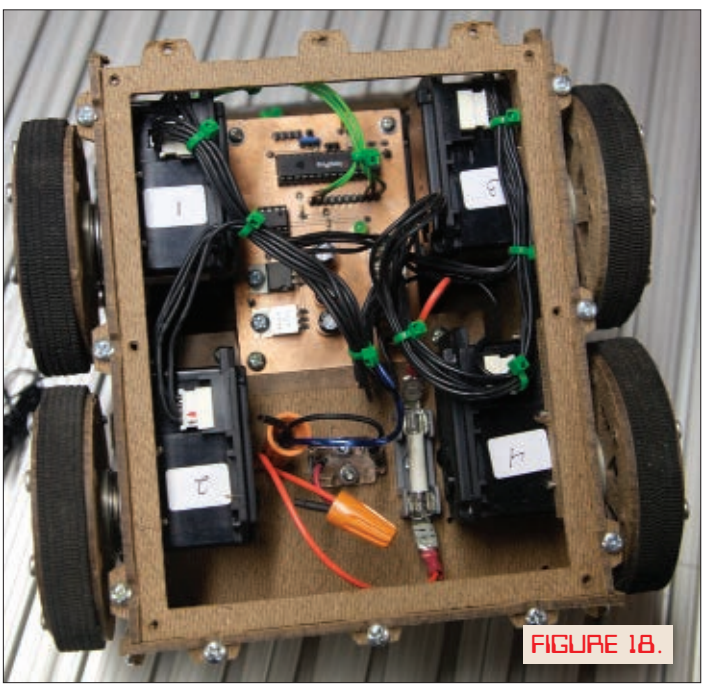


FIGURE 18.

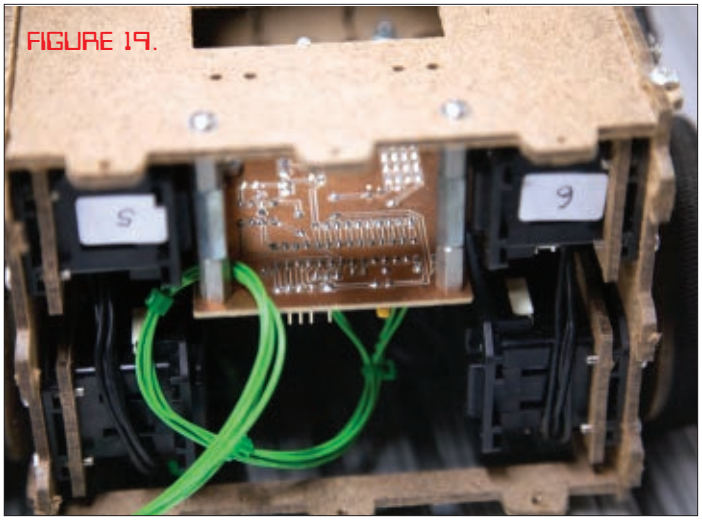


FIGURE 19.


FIGURE 20.

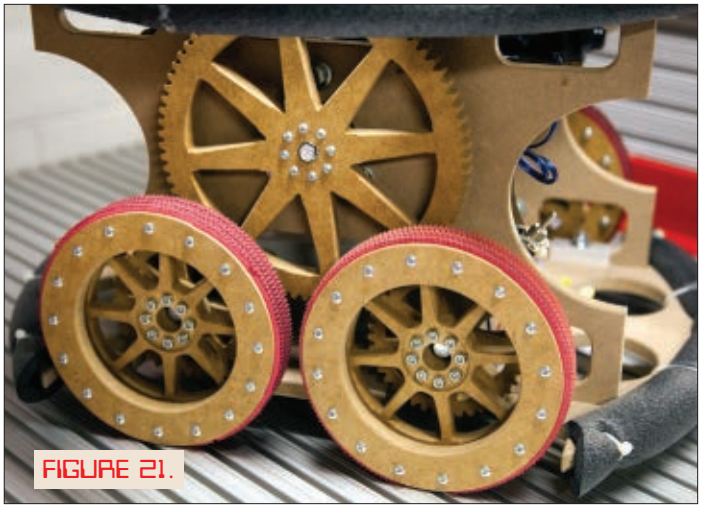


FIGURE 21.

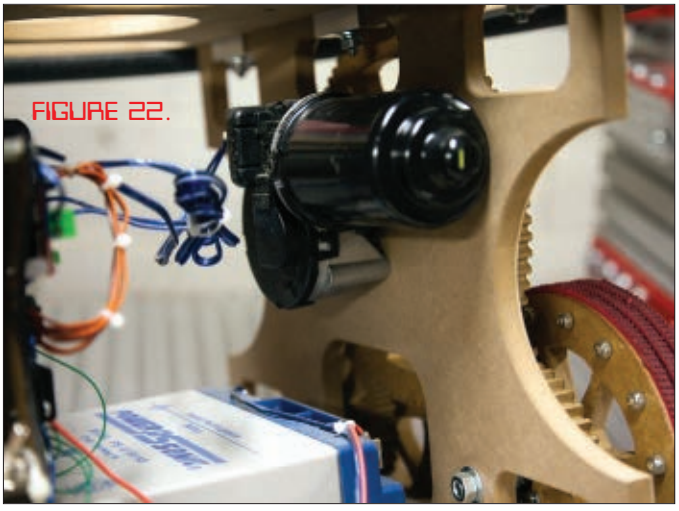


FIGURE 22.

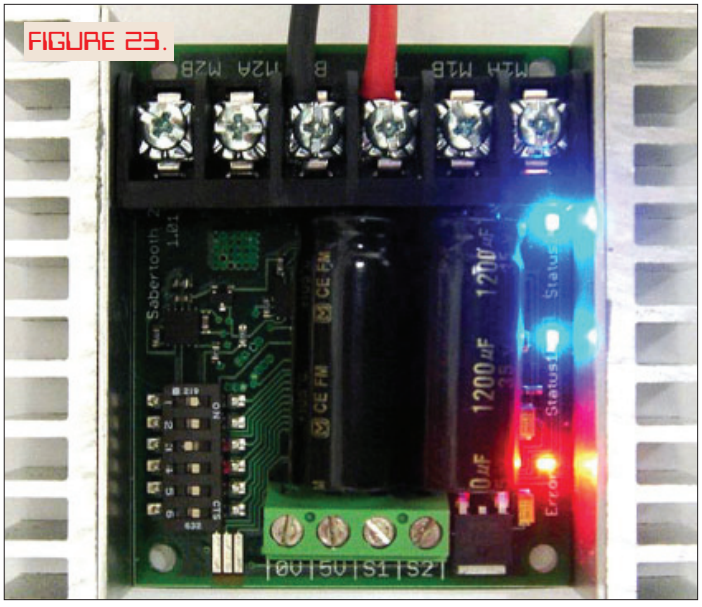


FIGURE 23.


FIGURE 24.

drive actuators and exposes the top of the PCB (printed circuit board).

The upper layer houses the flipper actuators and the mounts for the controller board as shown in **Figure 19**.

The design for these robots has held up surprisingly very well after many battles. See them in action at **www.youtube.com/watch?v=k1VnUkrzC2U**.

My ROS Robot

Figure 20 shows an RDS (Microsoft Robot Developer Studio) robot, and while it is not finished, I thought I would give you a preview.

The upper and lower platforms — as well as the sides — were cut on my KRMx01 CNC. On each side of the robot is a large gear (**Figure 21**) also cut on the CNC. This gear drives the two side wheels. The wheels and attached gears were assembled from components cut on the CNC, as well.

The RDS bot currently weighs in at over 50 lbs, so it needs motors with a lot of torque. I'm using two automotive windshield wiper motors (**Figure 22**) and an 18 Ah sealed lead acid battery. You need a heavy-duty motor controller to control this much power. The RDS robot uses a Sabertooth dual 25A motor driver (**Figure 23**) from Dimension Engineering.

The upper platform on the RDS is designed to hold a full-sized PC motherboard. It will also support a stand for a Microsoft Kinect camera.

A couple of videos of the RDS robot in action are available at **www.youtube.com /watch?v=VOH7AckxQWc.** and **www.youtube.com/watch? v=GzgmcDcgyuk.**

FIGURE 25.

Other Materials

Up to this point, I have discussed projects using MDF and hardboard. There are also many other materials that are well-suited to machining on the CNC: solid wood, plastics, composites, and even some metals. The E-chain shown in **Figure 24** is made from plastic and was cut on a CNC. The metal plate shown in **Figure 25** is made of aluminum and was machined on the KRMx02 CNC.

This video shows the KRMx01 cutting an aluminum hub used on the RDS robot: **www.youtube.com/watch? v=374ePHwPOlI.**

This video shows the KRMx01 cutting a polycarbonate panel used on one of the battle bots: **www.youtube.com/ watch?v=9svbNaP0KKw.**

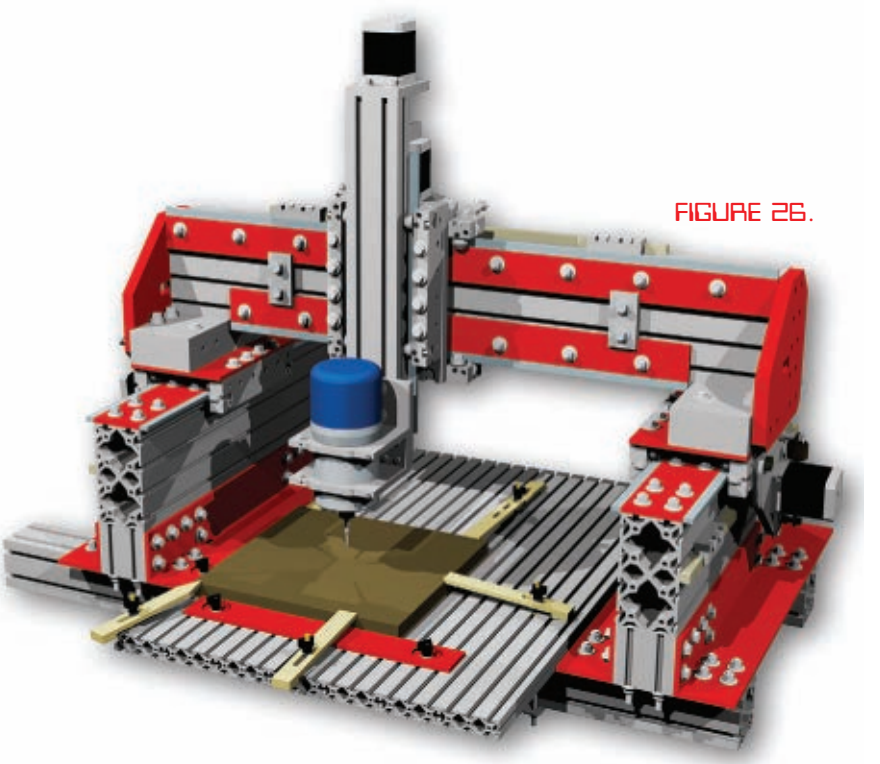


FIGURE 26.

Conclusion

What about Build 6? Since Builds 4 and 5 are currently being used in production using the KRMx02 design, I decided to build yet a smaller model that will be taken on the road for demonstrations. This machine (shown in **Figure 26**) will have an 18" x 14" cutting area and is small enough to be put on a stand with casters and rolled through any standard doorway. With all the parts in hand (shown in **Figure 27**), I can start this new journey.

Be sure to visit the Kronos Robotics website at **www.kronosrobotics.com** to see how it turns out. **SV**

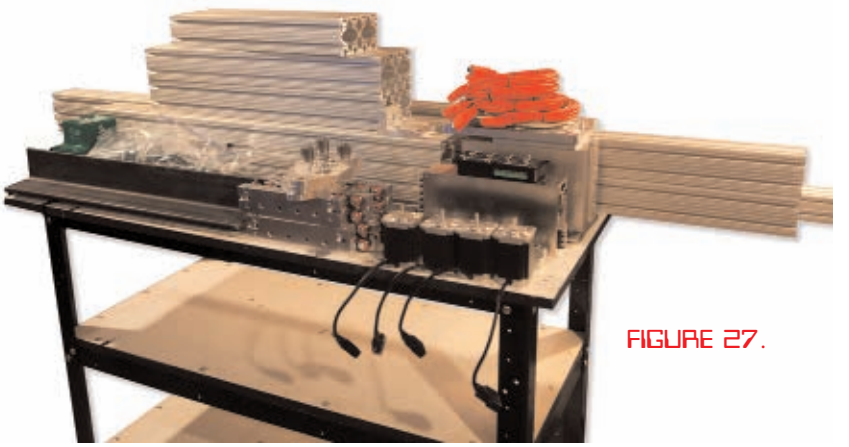


FIGURE 27.


Life with Robot!
Not a dream anymore
HerkuleX
DRS-0101
Dongbu Robot
HerkuleX
DRS-0201
Dongbu Robot
REAL SERVO
HerkuleX
"It's Rocky with robots...a heart-beat servo for everyone."
Dongbu Robot
Dongbu Robot Co., Ltd.
www.dongburobot.com
Headquarter : 11th Floor, Bucheon Techno Park 401, Yakdae-dong, Wonmi-gu, Bucheon-city, Gyunggi-do 420-734, Korea
TEL : +82-32-329-5551(ext.311), FAX : +82-32-329-5569, E-MAIL : robotsales@dongburobot.com
Factory : 27, 6 Gil, 4 Sandan, Jiksan-eup, Seobuk-gu, Cheonan-city, Chungcheongnam-do 331-814, Korea, TEL : +82-41-590-1700, FAX : +82-41-590-1701

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

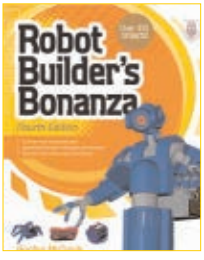
CD-ROM SPECIALS

NEW RELEASE

ROBOTICS

Robot Builder's Bonanza, Fourth Edition

by Gordon McComb



Robot Builder's Bonanza, Fourth Edition includes step-by-step plans for a number of motorized platforms. The book is described as a compendium of robotics topics, containing more than 100 projects, including 10 robot designs new to the fourth edition. These modular robots are low cost, and are made to be reproduced by readers with no training in mechanical construction.

$29.95*

Mechanisms and Mechanical Devices Sourcebook 5th Edition

NEW!

by Neil Sclater

Fully revised throughout, this abundantly illustrated reference describes proven mechanisms and mechanical devices. Each illustration represents a design concept that can easily be recycled for use in new or modified mechanical, electromechanical, or mechatronic products. Tutorials on the basics of mechanisms and motion control systems introduce you to those subjects or act as a refresher.

Reg $89.95 Sale Price $79.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts

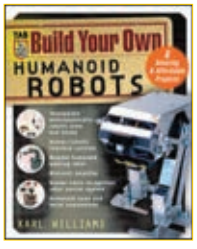
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

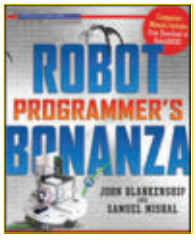


This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.

$24.95*

Robot Programmer's Bonanza

by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!

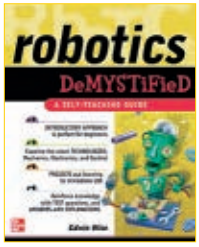
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

Save $10.00 Only $57.95!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, 09, 10 and 2011.

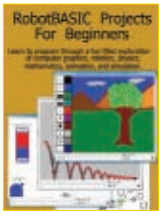
Any bot builders out there? Get cool robotics stuff from my store!

Call me at my order desk!

Visit my online store @ www.servomagazine.com

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal

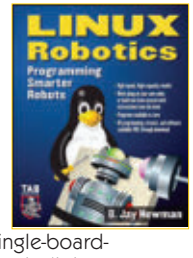


If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Price $14.95

Linux Robotics

by D. Jay Newman

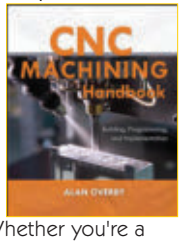


If you want your robot to have more brains than microcontrollers can deliver – if you want a truly intelligent, high-capability robot – everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

$34.95

CNC Machining Handbook: Building, Programming, and Implementation

by Alan Overby



The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

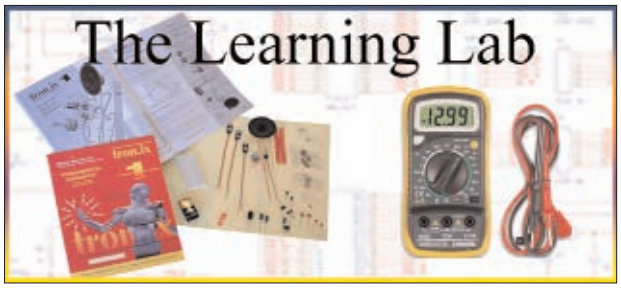
SPECIAL OFFERS

LEGO MINDSTORMS NXT Idea Book

by the Contributors to The NXT Step Blog

If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog – the authoritative online source for MINDSTORMS® NXT information and advice – has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

Reg $29.95 Sale Price $24.95



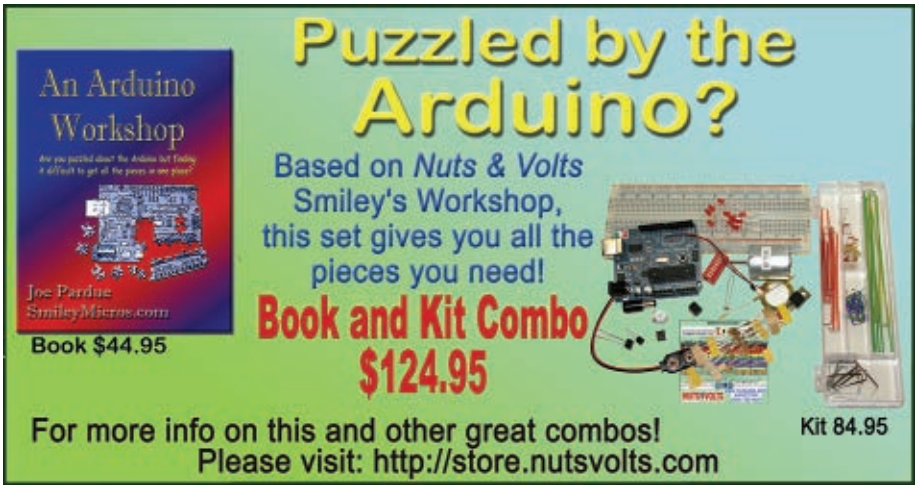
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

Regular Price $99.95 **Subscriber's Price $95.95**

Or order online
www.servomagazine.com

SPECIAL OFFERS



Forbidden LEGO

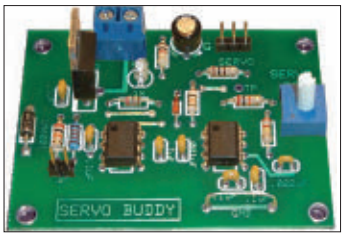
by Ulrik Pilegaard / Mike Dooley

Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse.

Reg $24.95 Sale Price $19.95

PROJECTS

The SERVO Buddy Kit

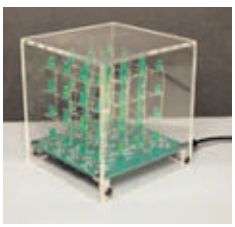


An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

3D LED Cube Kit

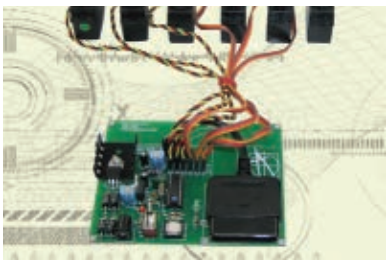


From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue.
Jig and plastic cases also available.

Subscriber's Price **$57.95**
Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*.
Assembled units available!
Subscriber's Price **$79.95**
Non-Subscriber's Price **$84.95**

Twin Tweaks

Bryce Woolley & Evan Woolley

THIS MONTH:

The Sensor Olympics

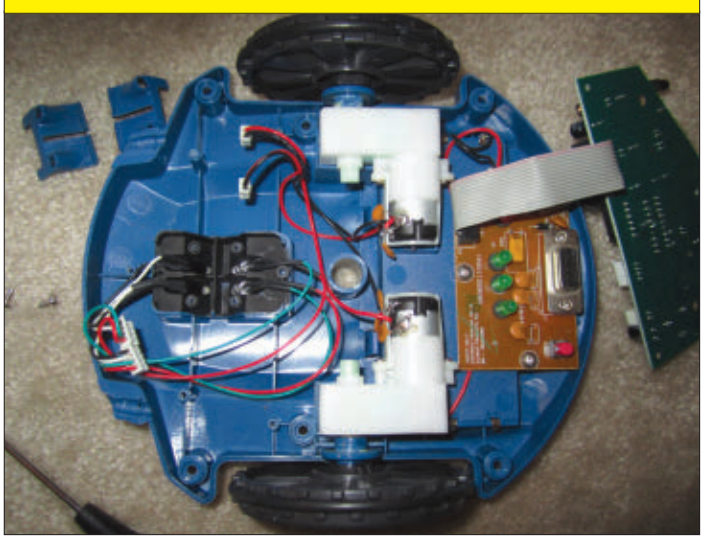
Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

The Scribbler from Parallax.

Spring is in the air, and all of the pent-up energy from a slumbery winter is ready to be released in a flurry of competitive sporting action. Two of our favorite sporting events are on the horizon: the Kentucky Derby in May, and the Summer Olympics in July and August. The Olympics are exciting — in part — because competitors come from all over the world, from a multitude of cultural and personal backgrounds, to compete in a series of events universally agreed upon to test skill and determination in various aspects of the human condition.

Robotics competitions are similarly thrilling but instead of a clash of physical prowess, designers pit ideas against one another. The spring and summer bring with them a host of robotics competitions, like the cosmopolitan RoboGames and the frenzied FIRST Robotics Competition. One design aspect of many robots that can mean the difference between a competitive edge and the status of also-ran is sensor selection. We've been lucky enough to work on such a broad range of kits that we've seen a plethora of sensors implemented in myriad ways, and we thought putting together a little Sensor Olympics would be a fun way to go beyond the datasheets and see how some sensors compared on a practical level.

Excavating the line following sensors.



The Thin Black Line

The first class of sensors we wanted to compare was comprised of sensors used for line following. Line following is as classic a robotics event as track and field races at the Summer Olympics, and as time honored a tradition as sipping mint juleps at Churchill Downs every May. A perfect blend of skill and speed, line following contests reward speedy drive trains as much as they do clever programming. The timelessness of the event has led to many solutions to the basic problem of tracking a line, and throughout our years we've gotten a general sense of some of the common problems faced by line followers. A devious plan involving a disappearing track and a light box began to take shape, but first we had to assemble a slate of diverse competitors.

Everyday I'm Scribblin'

Our first contender is the artistic ambassador from Parallax: the classic Scribbler robot. While the Scribbler

might be better known for drawing lines, it can also follow them using an array of sensors positioned at the front of its underbelly. The main webpage on the Scribbler declares the sensors that the Scribbler uses to follow lines to be line following sensors. These are likely not "line following sensors" per se – they do not examine the shape of the sensed image; disregarding filled circles and rhombi, and reacting only to lines.

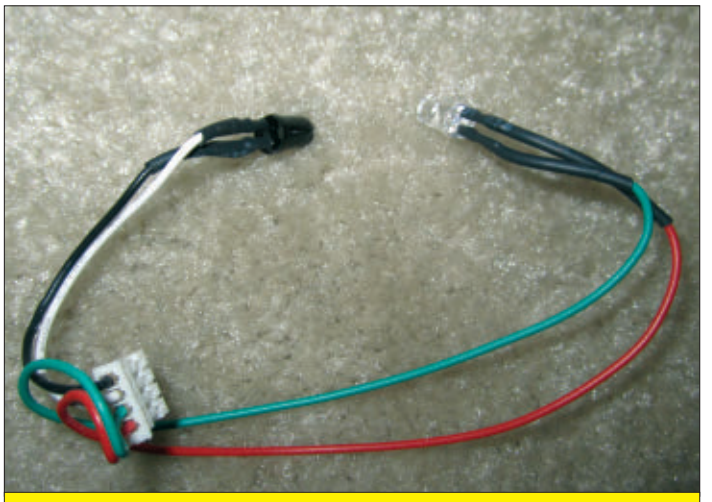
Many "line following sensors" are infrared sensors, able to sense the difference in infrared light reflected by white surfaces and black surfaces. A typical infrared line following sensor includes an IR transmitter that sends out a beam of infrared light, which then reflects from the nearby surface and is received by an IR receiver — usually situated adjacent to the transmitter. A white surface reflects more IR light than a black surface, and the intensity of the returning beam creates a voltage that can be interpreted to determine if the nearby surface is black (low voltage) or white (high voltage). This basic information is intuitively applied to the problem of following a black line on a white surface.

The Scribbler solves the line following problem with two side-by-side sets of IR emitters and receivers. One of the reasons why the line following challenge is so enduring is that there is such an intuitive connection between the information gleaned by the sensors and its logical implementation in the program. For the Scribbler, if the right sensor senses a black surface and the left sensor senses a white surface, you know that the line is on the Scribbler's right. The two sets of sensors allow you to keep track of where the robot is in relation to the line, and adjust its course accordingly.

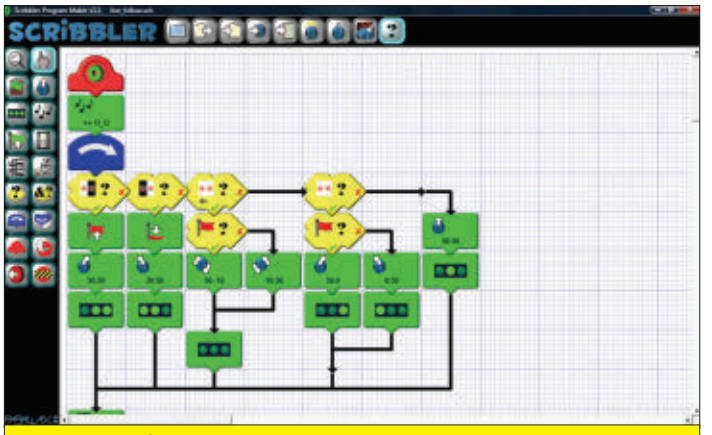
The schematic for the Scribbler (handily available on the Parallax website) labels the line following sensors as IR photo diode sensors. The humble photodiode sensors are accompanied by "signal conditioning and detection circuitry." The voltage generated by the received infrared beam on an IR detector is usually pretty small, so the signal conditioning circuitry likely includes an amplifier and perhaps a band pass filter to address noise.

To get a better look at the Scribbler's sensors, we popped the robot open. The sensors are buried deep in the bot, under the main circuit board and nestled away in a screwed together plastic housing. Once excavated, we saw that the sensors were your classic combination of an IR emitter and receiver. For the purposes of comparing the Scribbler sensors to the solutions used by other bots, we were hoping to find some sort of identifying marks that would lead us to a datasheet. Even though we were more interested in the practical performance of the sensors, the datasheet would still be a nice place to start.

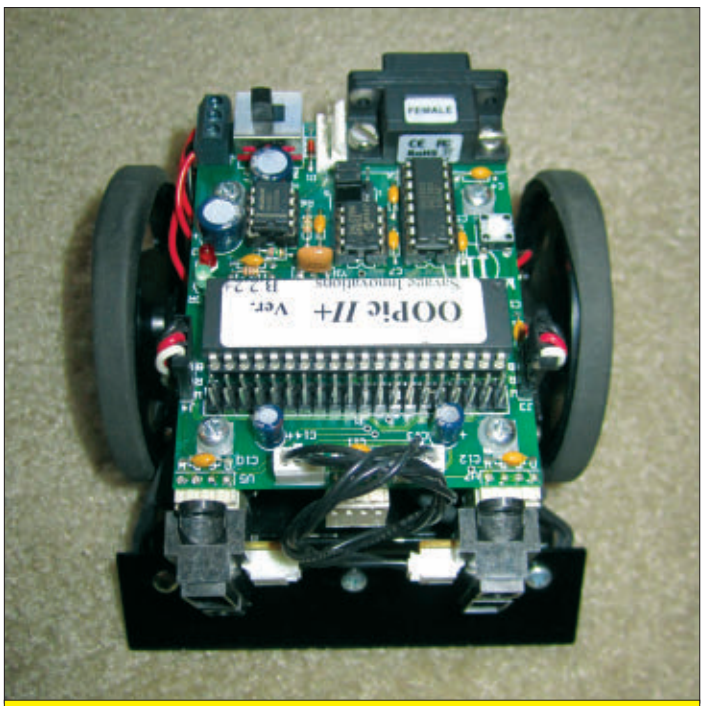
To get a general sense of the IR sensor landscape, we checked out the trusty Digi-Key catalog. Fairchild Semiconductor is a popular supplier of IR emitters and detectors, and the selection from Digi-Key shows that many of these IR emitters vary chiefly by their power output and peak wavelength. While these numbers might make for a nice comparison on paper, their connection to practical



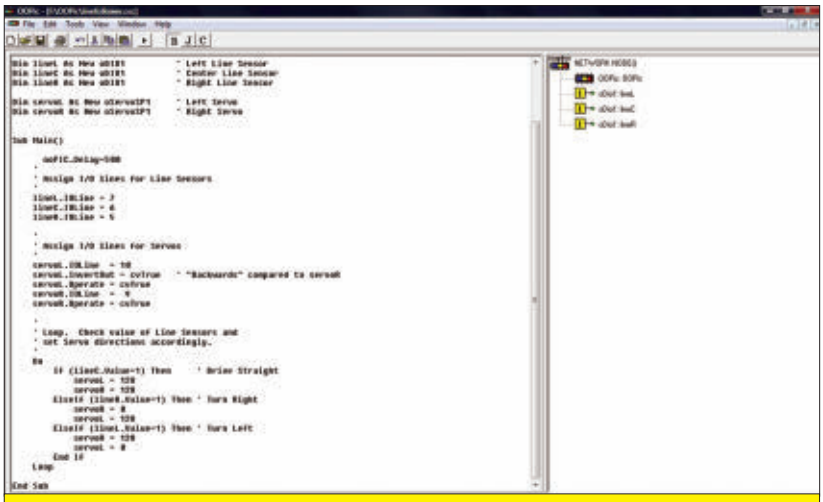
The Scribbler's phototransistors.



Scribbler line following program.

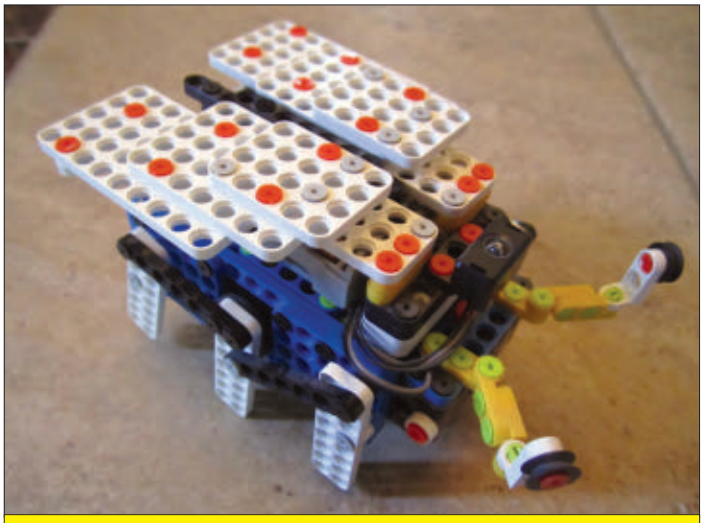


The ever-present Mark III.



Using OOPic to program the Mark III for line following.

The Fairchild sensors on the Mark III.



The OLLO Bug from Robotis.

performance is less than intuitive.

After reassembling the Scribbler, it was a simple enough task to program the line following behavior. Using the block-based programming from Parallax provided a colorful alternative to the normally highly textual exercise of programming. To ensure that we interpreted the Scribbler's behavior properly, we programmed the LEDs on the back of the bot to light up when the sensors saw the line. This way, we could distinguish scrupulous line following from lucky aimless wandering. With the Scribbler ready to go, all it needed was a few worthy opponents.

On Your Mark

One of our favorite recurring characters that would be sure to give the Scribbler a run for its money is the Mark III. Any self-respecting Sumo robot comes with the means to sense lines – the regulation Sumo ring is a filled black circle with a white outline. The Mark III accomplishes such a task with some sensors that we have crossed paths with before: the QRB1134 photosensors from Fairchild Semiconductor. In the January '09 issue, the Fairchild sensors made a prominent appearance as part of Evan's safe cracking robot. Implementing the sensors to track and reproduce a black and white test pattern was a task that was vexing at times, but with so many variables (and so little time) it was difficult to nail down how many problems could be ascribed to sensitivity issues and any other sensor shortcomings. In any event, the Fairchild sensors are affordable and largely reliable standbys that haven't led the Mark III astray in its many line following appearances in our column.

The Mark III uses three Fairchild sensors positioned under the robot's front wedge. The datasheet for the sensors is available on the Mark III webpage, and is filled with numbers that appear to provide a lot of technical detail but still leave something to be desired as far as practical implementation. Of course, the datasheet includes things like power requirements and all of the essentials for wiring up the sensors safely, but we were looking for information about soft implementation, in a sense. Where should the sensor be positioned? How does it handle noise?

One thing that seemed most readily applicable was a graph on the normalized collector current relative to the distance of the reflective surface. We actually reproduced a similar graph through our own tests for the safe cracking robot, and plotted output voltage against distance. The datasheet and our test confirmed that the Fairchild sensors actually have a decent ideal range, between about 2 and 15 mm.

We programmed the Mark III using OOPic. The logic of the program was a bit different than the one we loaded onto the Scribbler because the Mark III has three sensors instead of two. The program we used assumed that the center sensor would stay on the line, and then the sensor seeing the line after the middle lost track of it would determine the course correction. The Mark III was ready to go, and while a two robot duel is always fun, we thought one more competitor was required to make this Sensor Olympics event truly epic.

Bug Sport Redux

Despite the enduring popularity of the line following task, many of our other kits were chiefly interested in tasks like obstacle avoidance and did not have the means to face off against the Scribbler and the Mark III. Our field was filled, however, when we recalled the kits that one would expect to be most at home in a more pastoral kind of field: the OLLO Bugs from Robotis. We first encountered the OLLO Bugs in the April '09 issue, where we were introduced to the system's unique connectors and colorful personality. While the OLLO Bugs are simple kits meant to be accessible to a younger audience, the bots do have splashes of sophistication like a line following sensor.

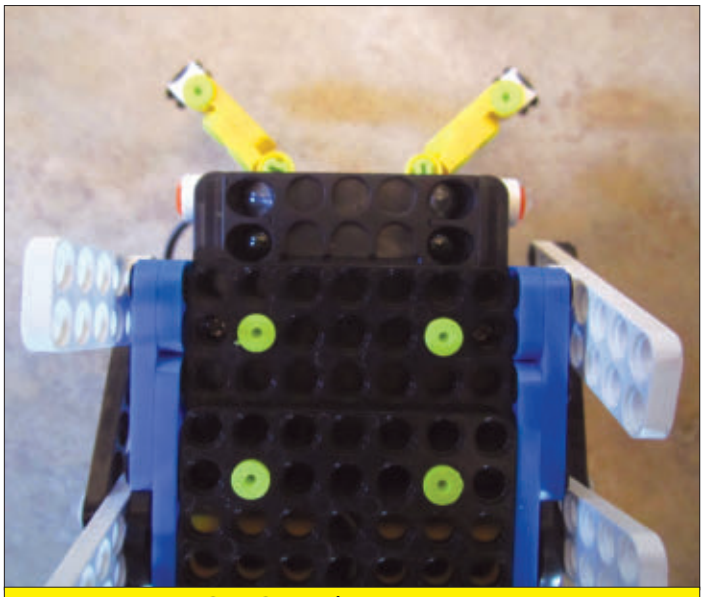
The OLLO Bug line following sensor is actually built into the control module, but a visual inspection reveals it to be comprised of the classic combination of IR emitter and detector in the same round packaging as the Scribbler sensors. Also like the Scribbler, the OLLO Bug relies on two sets of the emitter/detector pairs. Unlike the Scribbler, however, the IR sensors are somewhat exposed and far away from the surface they are sensing. The sensors peer out from the control panel like recessed lighting in a remodeled kitchen, but they are not as deeply ensconced as the Scribbler's sensors. Also, the OLLO Bug's sensors hover above the ground by about two inches — far higher than the low riding sensors on the Scribbler and the Mark III. We do think that the placement was determined more out of aesthetics than of unbridled confidence in the incorruptibility of the sensors. The control module happens to serve as the OLLO Bug's head, and so it is positioned about halfway up its body.

The OLLO Bug actually has a line following program preloaded into the robot. Interestingly enough, even though the OLLO Bug uses two sensors like the Scribbler, the logic of the program is different from both of its competitors. The sensors are meant to straddle the line and keep an eye on the white background. When one sensor sees the black line, the bug makes a course correction. The six-legged ambling provides a whimsical contrast to the smooth rolling of the Scribbler and Mark III, and with a field of three we were ready to put the line following sensors to the test.

Line Dancing

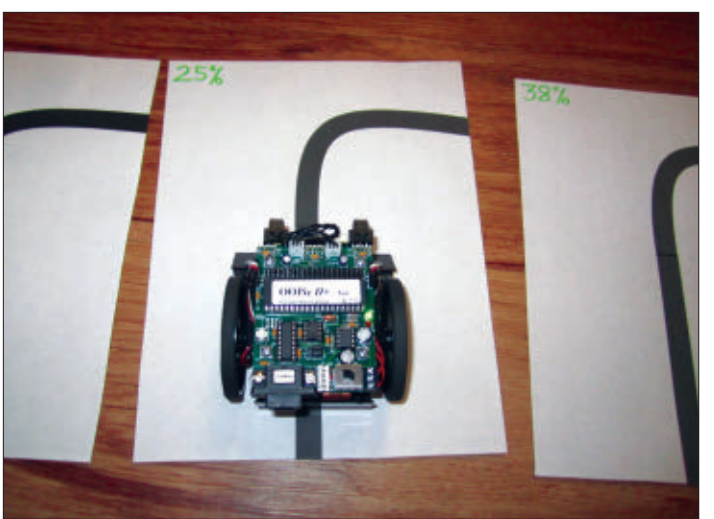
Now that we've been reintroduced to our line following kits, we can finally detect a problem that seems to be common to every line follower: the problem of contrast. In our experience, we've come under the impression that many line followers will become less and less cooperative when the contrast between the (usually black) line and the (usually white) background decreases. This intuitively makes sense because as the color of the surface changes, so does its reflectivity. But alas, none of the datasheets seemed to immediately suggest any such problem.

In many line following competitions, this is not a problem. The tracks are ensured to be a pitch black line on



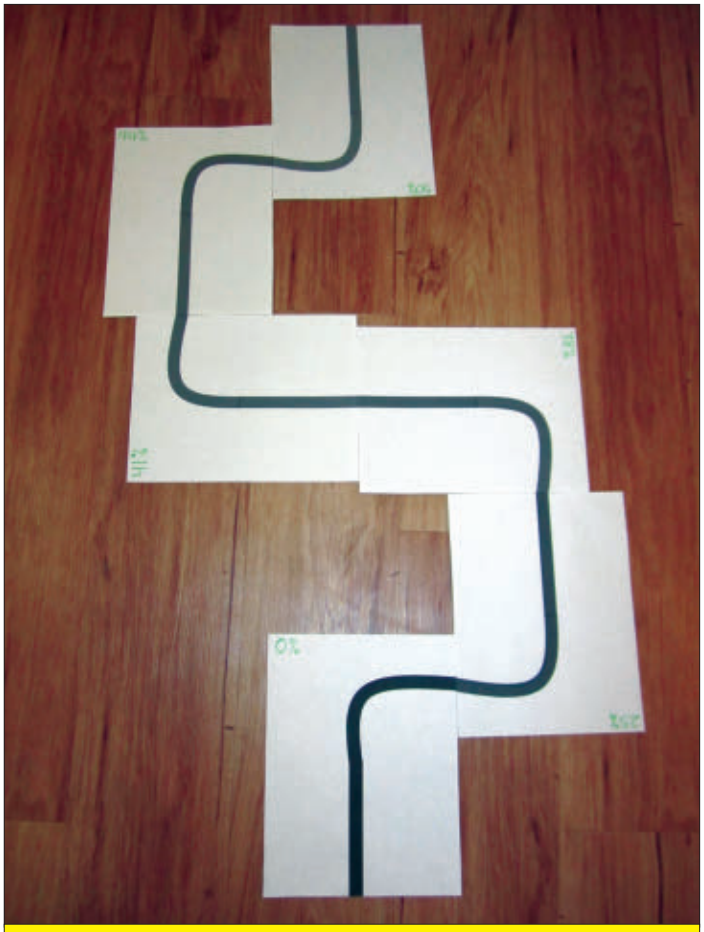
The OLLO Bug's infrared sensor.

The Scribbler follows a line instead of drawing one.



The Mark III makes its mark.

The OLLO Bug scrambles along.



The amazing disappearing track!

an immaculate background. In many circumstances though, the contrast might not be perfect. Maybe the background is off-white or smudged from use. Maybe the line has faded in the sunlight, or the printer was running out of ink. Seeing which sensors could best deal with less than ideal conditions seemed like an appropriate competitive test.

To test the effects of contrast, we engaged in a bit of a root finding exercise. We started with the classic black line on a white background – a straightaway ending in a curve that would ensure a successful journey wasn't just a fluke. We made a copy of that track, but adjusted the shape to have 50% transparency. In other words, we halved the contrast. We tested each robot on the 50% transparency line and — as expected — none of them could see it. In traditional root finding fashion, we took the halfway point of those boundary points for the next test – a line with 25% transparency. All three succeeded, so we went halfway between 25% and 50% — a 38% transparency line. All three succeeded again. In a couple more iterations, we were finally able to separate the best from the second best. At 41% transparency, the OLLO Bug and Scribbler could follow the line, but the Mark III could not. At 44% transparency, however, neither the Scribbler nor the OLLO Bug could detect the track.

Overall, we were impressed by the performance of all of the robots. Pushing 40% transparency is pretty impressive, and that sort of less-than-ideal contrast seems to be outside the realm of incidental track fading or smudging. But complacency is a killer, even in line following, so we devised another test meant to confound our field of competitors.

Lightboxing

Another test we wanted to perform dealt with ambient light interference. One of the advantages of using infrared sensors is that they don't depend on fickle visible light sources, which are subject to tons of variability. Ambient light, however, does carry some interference in the infrared spectrum. Household light sources emit infrared light that can potentially interfere with — among other things — conscientious line followers. Many IR sensors address this issue with signal conditioning circuitry like band pass filters. (The dark visor on the original Robosapien was meant as a defense against ambient light interference.)

So, how would we test ambient light interference? Ambient light would be most troublesome only if it shined directly on the IR sensors. Most IR sensors, however, are scrupulously hidden from view. The Fairchild sensors are shielded by the Mark III's plastic wedge, like a benevolent awning protecting loungers from the summer sun. The Scribbler's sensors are hidden deep within the belly of the bot, peeking through small openings as if looking through a key hole. The OLLO Bug sensors are the least protected of the bunch, but they're still out of direct line-of-sight from most potentially meddlesome light sources.

To get past all of these lines of cunning defense, we

would have to position our light source beneath the tracks. While this may sound like a task fit for M.C. Escher, we had a solution that was firmly within the realm of possible geometry: a fluorescent light box. The light box was the perfect size to accommodate the small robots and a segment of track. We performed the same root finding exercise as before, and it did indeed appear that the ambient light interference had an impact.

While under normal conditions, the Scribbler and OLLO Bug could follow the 41% transparency line, on the light box they were both as lost as a movie-goer falling asleep halfway between *Tinker, Tailor, Soldier, Spy*. Both had no problems with the 38% transparency line. The Mark III, however, could only follow lines up to a 25% transparency on the light box. It could likely do far better than that, but the moral of the story is that it had trouble following the grayest line that it could follow under normal conditions.

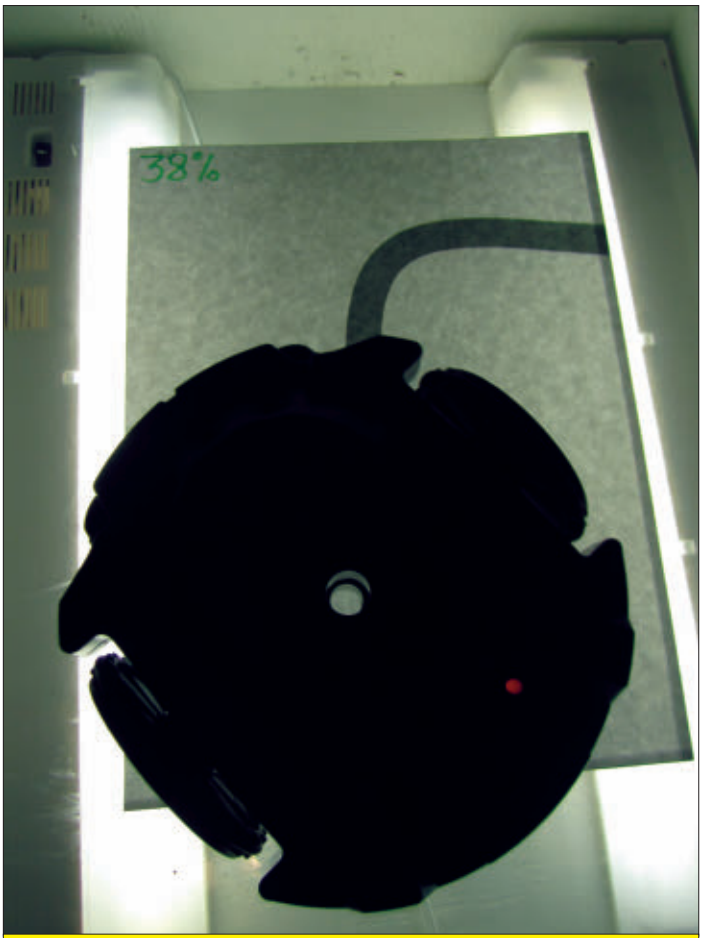
These tests are far from conclusive about the level of ambient light interference it takes to lead a robot astray. The newly discovered inconsistency in line following ability could be ascribed partially to other factors like declining battery life. In any event, it does appear that ambient light interference can indeed addle a line follower, and the robots are wise to keep their IR sensors protected as much as possible.

End of the Line

Our tests did not produce as dramatically varied results as we might have hoped, but it was enough to crown the winner and runners-up. Coming in third, we had the Mark III. We were a bit surprised that the intrepid bot was so easily bested by its competitors, probably because it is so solidly reliable in all of the other tests that we've used it for. The Fairchild sensors had irked us with some sensitivity problems with the safe cracking robot, mostly because they were quite needy regarding power requirements. Even so, the Fairchild sensors are trusty standbys that are wonderfully easy to implement and are apparently quite capable, barring a life or death situation involving a pale track on an illuminated floor like some sort of cruel game devised by the Master Control Program in the Grid.

Coming in second was the OLLO Bug. Even though it was able to follow the same tracks as the Scribbler, it seemed slightly less consistent and would occasionally wander off course. The OLLO Bug's finish seems fairly impressive given the fact that the placement of its IR sensors seemed driven more by aesthetics than by functional concerns.

The indomitable Scribbler came in first thanks to unshakable consistency, and because the debugging LEDs let us know that it really was sensing the line properly. Overall, these tests do have an important take-home lesson: The design of your sensor implementation really does matter! Especially for sensors without a lot of signal conditioning, protecting your IR sensors like a watchful mother goose can actually positively impact the robot's performance. Also, being careful to place the sensors within their ideal range helps minimize the vagaries of contrast and ambient light.



AMBIENT LIGHT TESTING.

AWARDING THE MEDALS.

There are a lot of sensors out there that will get the line following job done, but it is up to the sensible roboticist to implement them properly. We enjoyed getting reacquainted with many of these stellar kits, so be on the lookout for another event in the Sensor Olympics. SV

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Then and NOW

What Can Your Robot Do?

by Tom Carroll

Have you ever been standing beside your recently completed robotic creation — maybe at your robot club's exhibition or just out in your front yard —and somebody comes by and asks "What can your robot do?" You smile and tell them you've designed your robot to go over to your fridge, open its door, and retrieve a can of Pepsi and bring it over to you.

"Is that a hard task for a robot to accomplish?" they ask.

"Well," you tell them, "I'm using this Microsoft Kinect as my robot's eyes and ..."

"That thing can act like eyes? My kid has one of those with his Xbox video game. Is it that smart?"

You begin to tell them about your software and the SLAM programming techniques, and how the Kinect utilizes its two types of cameras and three microphones to see a person, and analyze and create a map from your easy chair to the kitchen and fridge.

They quickly change the subject ... "Uhhh, is that all it can do?"

"Well, no. It can navigate around my house and see different ..."

"Hey, that's nice. Umm, I noticed that you've got a nice green lawn, there. Well, I've go to go. See you around." They walk off ...

FIGURE 1. *Basic Basic* by James Coan.

FIGURE 2. The Altair computer in *Popular Electronics*.

Quite often, this is the average person's opinion or view of a robot or of people who build robots. Most people do not have a clue about the complex capabilities required by a robot to do even the simplest task.

I've written about how robots are built and what it takes to build one. I've even discussed how hobby robotics has changed over the years, but when it comes to that inevitable question — "What can your robot do?" — some people get a bit guarded and defensive about their pride and joy's capabilities. Probably the best answer to that question is simply, "Well, it makes me happy."

You've Built It. Now What?

It's late 1979 and your newly completed robot is sitting on your workbench, all charged up. Its 2708 EPROM has 60 lines of code burned in it that took you a week to program and de-bug. The 6502 microprocessor and 4K of RAM is perfect to make your machine do anything you so desire. "This is one cool robot!" you say to yourself. "The people at the computer club's robotics group will love Robbie."

You gently place Robbie on the floor, flip on the power switch, and hit the red 'reset' button. The robot emits a beep and slowly begins to crawl alongside one wall, its CdS photocell/IR LED emitter distance sensor carefully telling the processor to keep the robot 10 cm away from the wall. At the corner, it turns to the right and continues along another wall until it finds a doorway on the right. After a few seconds, it turns right and continues on into the next room.

The book you bought the year before — 'Basic BASIC,' 2nd ed by James S. Coan (shown in **Figure 1**) — sure helped you through the coding. BASIC stands for: Beginner's All-Purpose Symbolic Instruction Code. Very few experimenters use the original version any more. I actually used this book back in the early '70s as I slowly entered my lines of code into my AIM 65 computer that I'll describe a bit later.

Today's Robot Technology

Would you be interested in such a wall following robot today in 2012? You might be surprised that there are a lot of people who would enjoy building and programming a robot that could do this simple task using today's technology. CdS photocells are not used as much anymore, since experimenters implement emitter/photo transistor range finder modules instead. The 6502 microprocessor and 2708 EPROM are now considered museum pieces since robot builders prefer using the much cheaper microcontrollers available for their robots. Many beginner robots are capable of such programmed behavior and no more. Simple robot technology such as wall following or line following is critical in many robot designs from maze solvers to robot vacuum cleaners. The robot I just described is a great stepping stone to machines with higher levels of sophistication. It is easy for a robot designer to add more sensors and lines of code to allow their robot to accomplish more tasks.

Developing a Robot to Do What You Want It to Do

Many of the original PC users were asked that very same question back in the late '70s ... "What can your *computer* do?" Those were the days when Apple II computer users could develop spreadsheets with VisiCalc and word processing with MacWrite. Add in a few games and that was it.

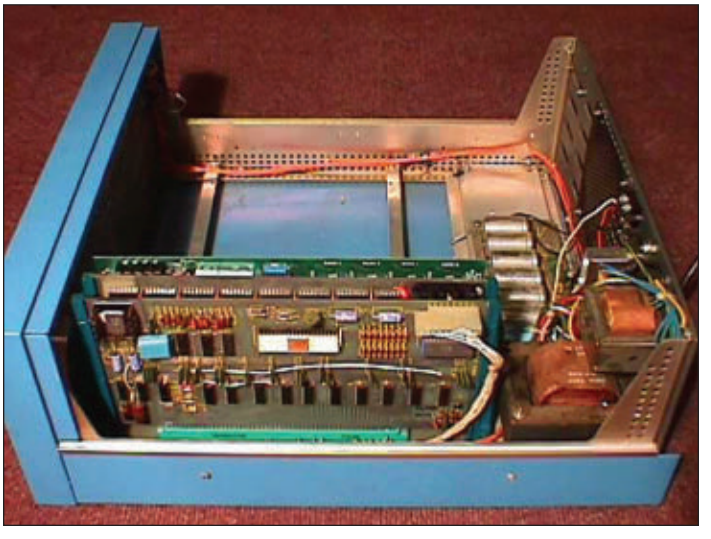
It was these types of programs that brought the microcomputer from a hobbyist's 'toy' to a useful product for the business world. So, if all that your latest robotic creation can do is carefully follow a wall, then turn and cover the next path out from the wall just maybe you can develop the next robot vacuum cleaner.

Don't Give Up On Your Robot Designs

We all know this story. Back in those early years of the microcomputer, two nerdy high school guys got together and developed a program to control city traffic patterns, and made a tidy profit of $20k for their work. One dropped out of school for a year to work at TRW. He decided to go back to school and later went to Harvard. His friend brought over a magazine one day in 1975 that had a computer kit featured on the cover (as shown in **Figure 2**). The computer kit company was asking readers to develop a good operating system for it and the two friends knew that they could build the program. The computer was the Altair 8800 shown in **Figure 3** — a $439 kit computer by the small Albuquerque company, MITS that contained the Intel 8800 microprocessor chip that cost almost that much alone

FIGURE 3. MITS Altair 8800 computer kit.

FIGURE 4. The Altair 8800 was well built.



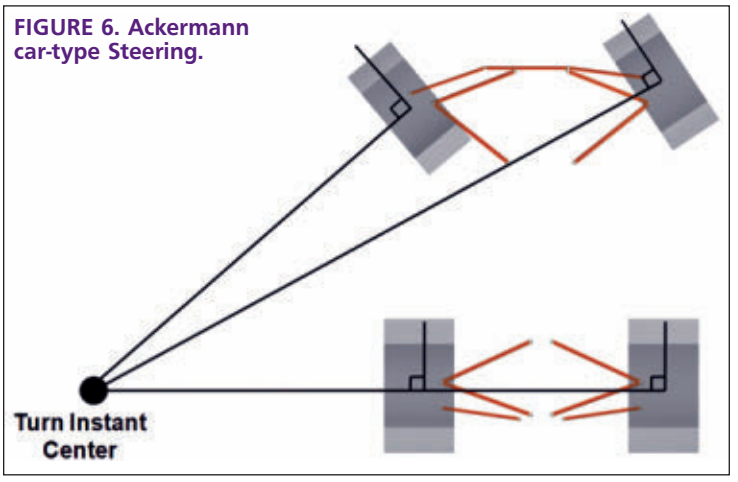


FIGURE 6. Ackermann car-type Steering.

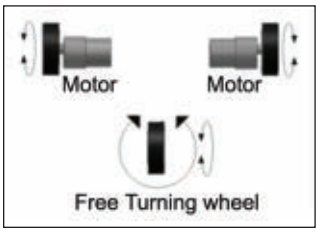


FIGURE 5. Differential steering concept.

just a few months prior. **Figure 4** shows the interior of the Altair with its roomy expansion area and beefy power supply.

The older of the two called the small company's owner, Ed Roberts, and told him they already had a program — which they really didn't. After a few frantic weeks of coding, the program was mostly completed. Ed invited the elder partner to come and demonstrate it while his friend stayed at school. The younger friend came out a bit later, and while living out of the Sundowner Motel near this company they tweaked the program over a few weeks while existing on take-out pizza and fast food. When the two friends demonstrated the program for the first time, it worked to everyone's amazement. They decided to call their new upstart Microsoft and based their fledgling company in a small office space near the computer company.

These two guys turned a hobbyist's toy into a real, workable computing system that changed the world as we know it. PCs today are far cheaper and more capable than the machines that existed in the early '70s. The same scenario could certainly happen to dedicated robot hobbyists turned serious designers. You can say to yourself, "This can be done." "People need it and will buy it." "I can make it happen." Just as these 'better' computers and the accompanying software were the result of a lot of hard work burning the midnight oil, the same applies to the best robot designs of today.

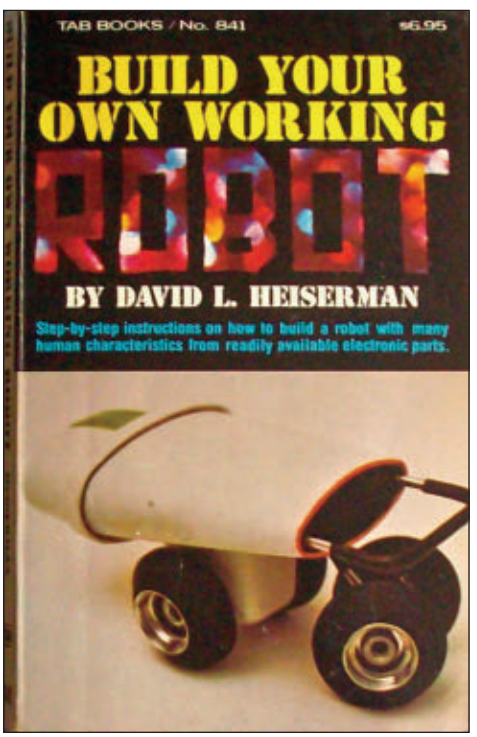


FIGURE 7. David Heiserman's *Build Your Own Working Robot* book.

Lay Your Groundwork First

It has been said that "a good design is 5% inspiration and 95% perspiration." Despite the previous scenario, don't start with the idea of making millions and billions with the greatest robot design ever. Work hard to make the best robot that you can. I would suggest that you remove any commercial aspirations from your potential designs and just concentrate on designing a good robot. Take helpful advice from experts and don't be put off by comments like "Is that all your robot can do?"

Most experimenters start out with basic capabilities, learn the mechanics, electronics, control methods, sensor types, and programming before taking the next steps. It really depends on how advanced you are in the field of robotics, your background, and most importantly in what type of robotics your desires and interests lie. You cannot always be in the right place at the right time, but you can develop the right product when the right time occurs.

If you're just starting into robotics, don't start with a complex robot design that will do everything. Most robot hobbyists start with some sort of mobile robot design. A high percentage of beginner (and advanced) robot designers use the differential steering method (see **Figure 5**) which is two fixed driven wheels or tank-type skid steering for their robot since the mechanics are easier to construct than for Ackermann steering (**Figure 6**) for car-type swivel wheels. Many builders use a BASIC Stamp or Arduino microcontroller with simple open-source programming initially. At this point in their robotics hobby, experimenters often go a bit further to develop their own designs in order to create

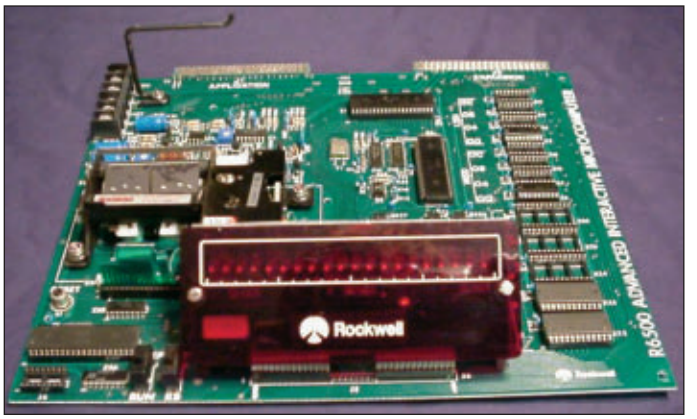


FIGURE 8. The Rockwell AIM-65 single-board computer.

a robot that does what *they* want it to do, so it's not just like everyone else's.

A Bit of History on Early Robot Designs

Jumping back a bit in time, the robot kits available to experimenters in the early '80s had a few unique capabilities when compared with today's machines. The 6502 wall follower I mentioned earlier was a pretty neat robot for its day, but sensors and processing power were limited. Turtle-style robots with a clear plastic shell covering the electronics (that also served as a bump sensor) were popular in the late '70s and early '80s. Heathkit designed a series of very capable robot kits that are still used today. RB Robot, Androbot, Tomy, and several other larger companies built kits or already assembled robots that were great platforms for early robot experimenters.

A book from 1976, *Build Your Own Working Robot* by David Heiserman (shown in **Figure 7**) and one from 1978, *How to Build a Computer-Controlled Robot* by Tod Loofbourrow helped to get many early robot experimenters started into the hobby. Heiserman's creation actually used a whole child's car as his robot base for the experiments, though the cover showed a robot that was not in the book. Tod's robot, 'Mike' used a pair of motorized wheels from a child's riding toy and a car-type lead acid battery for power, because it was a fairly large robot. The robot's intelligence was a 6502-based KIM-1 single-board, eight-bit computer — a classic experimenter's computer at the time.

Many experimenters of the early '80s jumped from being just PC users to becoming actual metal benders, and they built some great robots. When they were asked what their robot could do, the response was probably "It can follow a line drawn on the floor," or "It can follow my flashlight wherever I point it," or "It can follow a wall at a certain distance." Just as in our first scenario, the average onlooker was probably not that impressed. These types of robots were excellent beginning steps to learning basic robot design and programming. There were no cheap microcontrollers in those days; robot experimenters used the same eight-bit microprocessors that were the mainstay of the first PCs, such as the 6502, 6800, Z-80, and even the 8080, or ready-built single-board computers.

One of my first robots was based on the Rockwell AIM-65 shown in **Figure 8** — a single-board computer that had an alpha-numeric display, a full keyboard, and a tiny dot-matrix printer. As an employee of Rockwell, we could get these computers at a very good price. I believe we had the choice of 1K of RAM or for about $40 more, we could get a mighty 4K of RAM. Someone in one of our Texas facilities designed and built a bunch of really nice metal cases (shown in **Figure 9**) that had a tan finish and included a power supply so it could be used as a desktop computer. After buying the case, a floppy controller, updating the 20-character display to 40 characters, and adding a backplane for adding peripherals such as an old teletype, I ended up setting all those things aside and just used the AIM-65 by itself.

Trial and Error — Steps to More Advanced Robots

I've touched upon some old technology and robots that were starters for experimenters that could serve as basic platforms for more advanced robots. After you've designed and built a few different types of robots, the thought may come to you that you could possibly develop a product that others may want. iRobot started out back in the '90s with three MIT people and a lot of enthusiasm about robots. An oil well exploration robot and a pyramid-exploring robot were stepping stones to iRobot's later NexGen Multi-Function Floor Care commercial floor-cleaning machine (shown in **Figure 10**) which was part of a 1998 partnership with JohnsonDiversey (Johnson Wax). It was a stepping stone for their home Roomba robot vacuum cleaner project and also the later Scooba floor-washing robot.

iRobot's My Real Baby robot doll shown in **Figure 11** was a great idea but many adults found it to be a bit creepy (even though it was loved by most little girls). Marketed by Hasbro, the doll was fairly deep in the 'Uncanny Valley' with its many sensors, movements, and functions. It did not last long in the marketplace.

Most companies have occasional 'off' products, but use these experiences as a valuable learning tool. Hobbyists and home robot experimenters should follow this same helpful advice.

FIGURE 9. The AIM-65 in a convenient case.

FIGURE 10B. iRobot's Autoclean.

FIGURE 10A. The iRobot NexGen multi-function floor care machine.

FIGURE 11. My Real Baby robot doll.

FIGURE 12. The affordable Hokuyo laser range finder.

FIGURE 13. The LMS200 range finder from SICK.

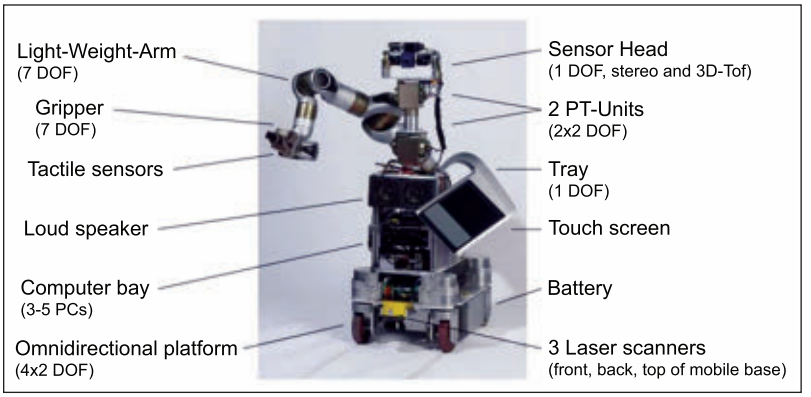


FIGURE 14. Care-O-Bot's many functions.

What Type of Robot Do You Want?

When developing that special robot design that will accomplish a particular task, you need to look at the bigger picture in the same way that Bill Gates and Paul Allen did 35 years ago. With the introduction of Intel's new 8000 series of microprocessors, they saw the possibility of a truly functional PC as a viable market for their talents as software developers. Today, you might look at the recently-introduced Microsoft Kinect for the Xbox 360 as the ideal vision system for a robot design that you have in mind. Robot builders have literally jumped onto this video gamer's device as a turning point in functional mobile robot designs.

Needless to say, there are numerous styles and types of robots. Mobile ground vehicles, aerial robots, sea surface and under sea robots, industrial and personal robots, household, and medical robots are just a few of the more popular categories. Each of these systems has various configurations. Robots can be autonomous or tele-operated, or a combination of both. The many types of sensors available today can be used on robots to allow for capabilities that were unheard of just a few years ago. The $130 Kinect has replaced the very popular $1,100 Hokuyo URG-04LX-UG0 shown in **Figure 12**, and the $5,000+ SICK LMS20 shown in **Figure 13** has become the laser range finder of choice for many high-end experimental robots. These newer types of sensors can certainly add many enhancements to robotic projects.

Just as the millions of apps and software packages available for today's smart tablets, phones, and computers give these devices almost unlimited capabilities, many people feel that it is programming and software that give our robots their abilities. Software such as RDS and ROS are just a few of the more powerful systems that are available to robot experimenters today. Simpler programs for use with microcontrollers such as P-Basic developed for the Parallax BASIC Stamp series, RobotC for the popular LEGO and VEX microcontrollers, Tiny Basic, Robot Basic, C, C++, and others have a lot of the functionality of more complex programs.

Final Thoughts

When that day comes and you are asked — "What can your robot do?" — just smile and say, "I'll show you." It doesn't have to be a robot like Willow Garage's PR-2 or the Fraunhofer Institute's amazing Care-O-Bot shown with its many functions in **Figure 14**. It just has to do what *you* designed it to do and nothing more. SV

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

Advertiser Index

DF RobotShop Rover

Compatible with

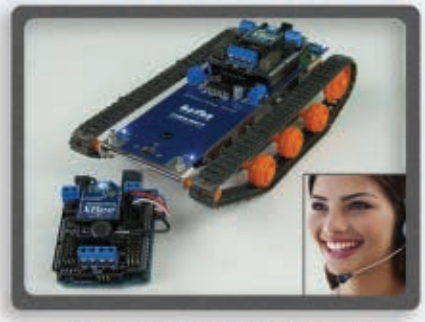
ARDUINO

MYROBOTS.COM
Connect your Robots. Reap the benefits™

Affordable, Flexible and Scalable

Compatible with thousands of off-the-shelf parts!

Wireless Control



Speech Control*

Mecanum

RobotShop
Robotics at your service!
www.robotshop.com

Scan on your smart-phone with a bar reader app

ADDITIONAL VARIANTS TO COME
THE EVOLUTION CONTINUES...

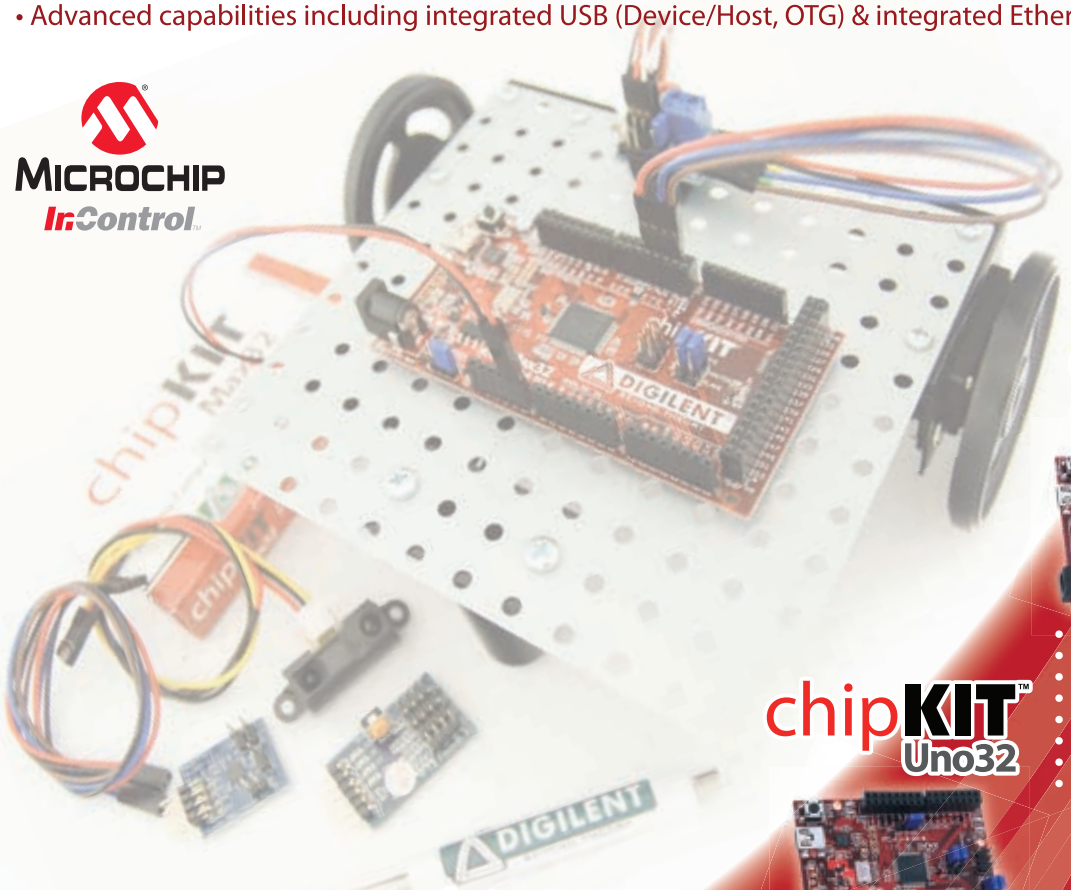
chipKIT™

Advanced 32-bit MCU Power for the Arduino™ Community

chipKIT™ development boards are the first 32-bit-microcontroller-based platforms that are compatible with many existing Arduino™ code examples, reference materials and other resources. They can be programmed using an environment based on the original Arduino™ IDE modified to support PIC32.

- Pin-out compatibility with many existing Arduino™ shields that can operate at 3.3V
- Lower price-point at four times the performance than existing solutions
- Advanced capabilities including integrated USB (Device/Host, OTG) & integrated Ethernet

MICROCHIP
InControl



chipKIT™ Max32

- Microchip® PIC32MX795F512
- 80 Mhz 32-bit MIPS
- 512K Flash, 128K RAM
- USB 2.0 OTG controller
- 10/100 Ethernet MAC
- Dual CAN controllers
- Arduino™ "Mega" form factor
- 83 available I/O

chipKIT™ Uno32

- Microchip® PIC32MX320F128
- 80 Mhz 32-bit MIPS
- 128K Flash, 16K SRAM
- Arduino™ "Uno" form factor
- 42 available I/O

chipKIT™ Network Shield

- 10/100 Ethernet
- USB Host, Device, OTG
- Dual CAN transceivers
- Dual I²C™ connectors
- 256kbit I²C™ EEPROM

chipKIT™ Basic I/O Shield

- 128x32 OLED Graphic Display
- Digital temperature sensor
- 256kbit EEPROM
- 4 switches, 4 push buttons, 8 LEDs
- 4 Open drain transistor outputs
- Analog potentiometer

DIGILENT®
BEYOND THEORY

www.digilentinc.com/chipkit